TRABAJO FINAL DE CARRERA

TÍTULO DEL TFC: Acceso a servicios Web que han sido publicados en una red P2P DHT mediante dispositivos móviles

TITULACIÓN: Ingeniería Técnica de Telecomunicaciones, especialidad Telemática

AUTOR: Javier Olivares Sierras

DIRECTOR: Dolors Royo Vallés

Fecha: 14 de Noviembre de 2007

Título: Acceso a servicios Web que han sido publicados en una red P2P DHT mediante dispositivos móviles

Autor: Javier Olivares Sierras

Director: Dolors Royo Vallés

Fecha: 14 de Noviembre de 2007

Resumen

El objetivo principal de este proyecto es poder acceder a servicios Web publicados en una red P2P-DHT mediante dispositivos móviles. Esta red es creada por una API de Java denominada FreePastry, la cual sigue una estructura lógica de anillo con la capacidad de administrarse para aumentar su eficiencia.

Sobre esta API, el alumno Roger Martínez Terés creó una aplicación para lograr la publicación de servicios Web alojados en servidores Apache – Tomcat. Nuestra tarea ha consistido en extender este proyecto haciendo posible acceder mediante dispositivos móviles a los servicios Web publicados en una red P2P-DHT.

Para ello, hemos estudiado distintas posibles soluciones de las que hablaremos a lo largo de la memoria. Sopesamos las opciones que consideramos más efectivas y, finalmente, proponemos una solución basada en los servicios Web creados sobre la plataforma .NET de Microsoft.

La plataforma .NET de Microsoft nos proporciona transparencia de redes y una buena interfaz gráfica, facilitando la utilización de la aplicación inicial a usuarios inexpertos. Todo esto sin perder la funcionalidad inicial del sistema original, es decir, la posibilidad de publicar servicios mediante Tomcat + Axis, proceso que también detallaremos.

Con la realización de este proyecto, hemos logrado extender una red robusta, distribuida y eficiente sobre la que colgar servicios útiles a una comunidad conectada mediante redes locales (como puede ser una empresa o universidad), con un empleo muy sencillo y accesible para cualquier usuario. A todo esto se suma la independencia de la plataforma para que el sistema funcione gracias al software multiplataforma que utilizamos.

Title: Acces to Web services that have been hosted in P2P – DHT web by mobile devices

Author: Javier Olivares Sierras

Director: Dolors Royo Vallés

Date: November, 14th 2007

Summary

The main goal of this project is to access to Web services published in a P2P-DHT web by mobile devices. This web is created by a Java's API called FreePastry, that follows a logical ring structure capable of self-administration to improve it's efficiency.

On this API, student Roger Martínez Terés made an application that allowed to publish Web services hosted in Apache-Tomcat servers. Own task has been to extend this project making possible to access by mobile devices to Web services published in a P2P – DHT web.

For that, we have studied different possible solutions that we will speak through this document. We weighed up the options that we considered more effective and, finally, we suggest a Web service based solution created on .NET platform of Microsoft.

.NET platform provides web transparency and a good graphical interface, making easy the use of the initial application to non-expert users. All without losing the initial hability of the original system, that is, the possibility to host services through Tomcat + Axis, process that will be detailed.

Finally we've managed to create a solid, distributed and efficient network on wich to upload useful services to a conected through local network's community (like it could be a company or university), with a very easy use and accessible to any user.

Índice

CAPÍTULO 1: INTRODUCCIÓN 1
1.1 Razón y Oportunidad del Proyecto 1
1.2. Objetivos 2
CAPÍTULO 2: CONCEPTOS BÁSICOS 1
2.1. Introducción a los servicios Web 1
2.1.1. XML 2
2.1.2. SOAP 3
2.1.3. WSDL 3
2.1.4. UDDI 4
2.2. P2P-DHT 5
CAPÍTULO 3: SOFTWARE UTILIZADO 8
3.1. Java y JVM 8
3.2. C# y .NET framework 8
3.3. Axis 9
3.4. Tomcat 9
3.5. FreePastry 9
3.5.1. Estructura de la red Pastry 9
3.5.2. Encaminamiento 10
CAPÍTULO 4: PREPARACIÓN DEL ENTORNO DE TRABAJO 12
4.1. Instalación de software 12
4.1.1. JVM 12
4.1.2. .NET Framework 12
4.1.3. Tomcat 12
4.1.4. Axis 13
4.1.5. Instalar aplicación Web PastryWebMovil 13
4.1.6. Instalar FreePastry 13
4.2. Creación de un servicio Web 14
CAPÍTULO 5: APLICACIÓN EN FREEPASTRY 16
5.1. Aplicación en FreePastry 16
5.1.1. Ejecución de la aplicación 16
5.1.2. Utilización del menú 16
5.2. Mensajes en la red P2P 19
5.2.1. Mensaje de publicación 20
5.2.2. Mensaje para añadir servicios a la lista 20
5.2.3. Mensaje para pedir la lista 21
5.2.4. Mensaje de búsqueda 22
5.2.5. Mensaje de consumición 22
CAPÍTULO 6: DISEÑO DE LA APLICACIÓN EN .NET 24
6.1. Sopesando opciones 24
6.2. Aplicación Web PastryWebMovil 26
6.2.1. Comunicación 26
6.2.2. Utilización 27
6.2.3. Servicio Web EncenderApagar 29
CAPÍTULO 7: PLAN DE TRABAJO 31
7.1. Tareas 31
7.2. Costes asociados a este proyecto 32
CAPÍTULO 8: CONCLUSIONES 33
8.1. Objetivos asumidos 33

8.2. Posibles mejoras 33
8.3. Ampliaciones futuras 33
8.4. Impacto ambiental 33
8.5. Conclusiones personales 34
CAPÍTULO 9: BIBLIOGRAFÍA 35
9.1. Recursos Web 35
9.2. Libros 35
ANNEXOS 36
ANEXO 1. WSDL DEL SERVICIO CALCULADORA 37
ANEXO 2. CÓDIGO FUENTE APLICACIÓN PASTRY 40
2.1. MyMain.java 40
2.2. MyApp.java 45
2.3. MyMsgXML 55
2.4. WebServiceP2P.java 58
2.5. SOAPClient.java 60
2.6. Parse.java 62
2.7. Escuchar_multicast.java 65
ANEXO 3. CÓDIGO FUENTE PASTRYWEBMOVIL 67
3.1. MobileWebForm1.aspx.cs 67
3.2. MulticastSearch.cs 72
ANEXO 4. CÓDIGO FUENTE ENCENDERAPAGAR 75
4.1. MobileWebForm1.aspx.cs 75
4.2. TestMulticastOption.cs 77

CAPÍTULO 1: INTRODUCCIÓN

En este capítulo se describen cuales son los objetivos del proyecto y como se enmarca éste en el mundo actual.

Después de la introducción, esta memoria se inicia con la explicación de una serie de conceptos básicos para poder tener una visión acertada de lo que se explicará en el resto de capítulos.

En el siguiente capítulo mencionaremos el software que hemos utilizado para desarrollarlo y poder ejecutarlo, así como la utilidad de este software y porqué lo hemos seleccionado.

En el cuarto capítulo explicaremos paso a paso que ha de hacer un usuario para instalar el software requerido y detallaremos que se está haciendo en cada paso para tener listo el entorno de trabajo.

En el quinto capítulo iniciaremos la explicación de cómo funciona la aplicación que extiende la funcionalidad de FreePastry.

Después de eso detallaremos el diseño de nuestra aplicación para comunicarnos con la red Pastry y poder acceder a los servicios Web dedicados a dispositivos móviles.

En el séptimo capítulo explicaremos el plan de trabajo que hemos seguido durante estos meses para la realización de este proyecto.

Finalmente expondremos nuestras conclusiones sobre este trabajo y sus posibilidades futuras.

1.1 Razón y Oportunidad del Proyecto

Las telecomunicaciones tienen una penetración absoluta en el mundo empresarial y docente. Resulta impensable el no disponer de una red de área local en cualquier tipo de centros con la que mantener comunicados a empleados y aplicaciones. De la misma manera también es impensable el no poseer conexión a Internet y sus servicios Web en entidades con un mínimo de investigación y desarrollo.

Por otra parte, en el mundo de Internet las redes centralizadas cada vez tienen menos cabida para dar paso a sistemas mucho más robustos, escalables y eficientes: las redes P2P –DHT. Como ejemplo tenemos programas tan conocidos como eMule o Kazaa, orientados a compartir archivos entre una multitud creciente de usuarios, y todo ello evitando la saturación gracias a su sistema distribuido. Otro ejemplo de este tipo de redes es la tecnología FreePastry, de la cual nos valdremos en este proyecto para alcanzar nuestros objetivos.

Nuestro proyecto combina la utilización de servicios Web personalizados y accesibles por cualquier dispositivo con la eficiencia de las redes P2P – DHT. Todo con la máxima transparencia posible para que cualquier usuario pueda acceder y utilizar estos servicios de la forma más intuitiva posible.

De esta forma, un administrador de redes puede crear un servicio Web, publicarlo y ponerlo inmediatamente en manos de empleados o usuarios que no necesitan nociones de informática para utilizarlos. Si estos servicios son accesibles desde tecnología móvil, se facilita así el acceso a empleados en continuo movimiento.

1.2. Objetivos

Este proyecto se divide en varias partes que describiremos durante este trabajo, pero sus objetivos principales son:

1. Desarrollo de una aplicación que nos permita acceder a servicios Web mediante dispositivos móviles. Para realizar esta operación estudiamos diversas tecnologías hasta decantarnos por la plataforma .NET de Microsoft por razones que se detallaran a lo largo de esta memoria.

   Este objetivo ha sido alcanzado puesto que hemos realizado pruebas con simuladores y posteriormente con dispositivos reales, accediendo a nuestros servicios Web.

2. Integración de esta aplicación en un entorno desarrollado en otro proyecto final de carrera, “Publicación y acceso a servicios Web en una red peer to peer DHT”, la cual nos permite acceder y ejecutar servicios Web alojados en servidores Axis-Tomcat, previamente publicados en una red P2P-DHT basada en la tecnología FreePastry.

   Esto también lo hemos logrado ya que hemos conseguido acceder a servicios publicados en la red Pastry mediante aplicaciones Web creadas por nosotros.

Para la evaluación y análisis del proyecto se propone como objetivo secundario el desarrollo de algún servicio Web adicional con el que comprobar que el sistema realmente funciona en su totalidad. Para ello hemos creado un servicio simple denominado EncenderApagar que enciende y apaga ordenadores de forma remota.

CAPÍTULO 2: CONCEPTOS BÁSICOS

En este capítulo detalla los conceptos clave de este proyecto, como son los servicios Web y las redes P2P – DHT.

2.1. Introducción a los servicios Web

El desarrollo y evolución de los servicios Web responden a la necesidad de crear una comunicación entre aplicaciones independientemente de la plataforma que las sostiene. Esto se logra gracias a estándares, reglas y protocolos comunes entre todos ellos.

Las comunicaciones y la informática tienden a facilitar su uso a usuarios inexpertos, y a realizar operaciones complejas de la manera más transparente posible. Gracias a que estos estándares y protocolos se basan en texto, el manejo de estos servicios es cada vez más intuitivo y sencillo.

Además los servicios Web se comunican por Internet mediante protocolos no bloqueados por firewalls, como es http, el cual se conecta a los equipos a través del puerto 80, siempre abierto para posibilitar la navegación.

La simplicidad de esta transacción de información entre equipos hace que los sistemas puedan ser distribuidos. Los sistemas distribuidos son, como su propio nombre indica, servicios ofrecidos por varios equipos que conjuntamente crean un sistema. Las ventajas son muchas, pues no depender de un único equipo añade robustez y capacidad de procesado, además del ahorro económico que supone (es mas barato comprar tres ordenadores que comprar uno capaz de hacer el trabajo de tres).

Por todo esto las nuevas tendencias están haciendo evolucionar los servicios Web de forma que las aplicaciones no tengan porque estar en el equipo del usuario, y centralizar su utilización en servidores. Gracias a ello las mejoras y actualizaciones en dichas aplicaciones son puestas en manos del usuario con mucha más penetración y eficiencia.

Para comprender el funcionamiento de los servicios Web es necesario conocer los estándares sobre los cuales se fundamentan, que actualmente son SOAP, XML, UDDI y WSDL principalmente.

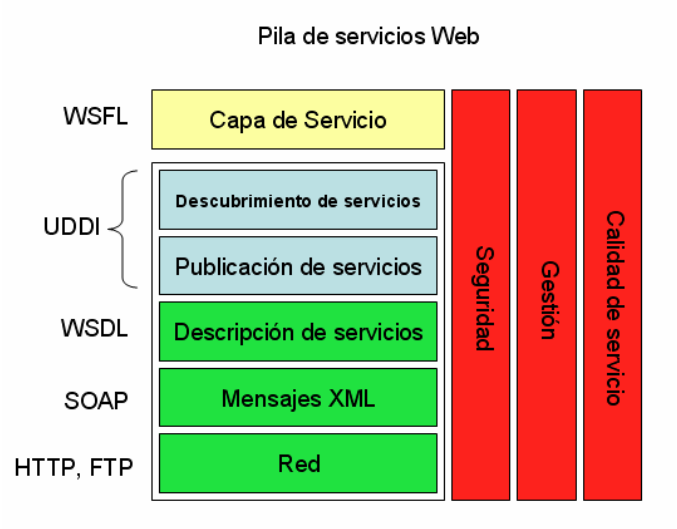

Fig. 2.1 Pila de Servicios Web

2.1.1. XML

XML (eXtensible Markup Lenguage) es un lenguaje textual basado en etiquetas y que define las normas para la creación de otros lenguajes según las necesidades inmediatas.

En XML podemos definir nuestras propias etiquetas para enmarcar los datos a mostrar, preocupándonos en segundo plano del aspecto estético de la información. Además, tiene una sintaxis muy estricta, propiciando la comunicación entre máquinas y ofreciendo una información estable a la hora de procesarla y tratar con ella.

Otra ventaja es que gracias a su gran extensibilidad no requiere de nuevas versiones para funcionar en futuros navegadores, simplemente basta con aplicar un validador de documentos para su adaptación. El validador se encarga de decidir si la sintaxis empleada en el documento es correcta. A continuación mostramos un ejemplo de documento XML:

```
<?xml versión=”1.0” encoding=”UTF-8”?>
<alumnos>
        <alumno identificacion=1>
                <nombre>Juan</nombre>
        </alumno>
        <alumno identificacion=2>
                <nombre>Pepe</nombre>
        </alumno>
</alumnos>
```

En este ejemplo podemos ver como inicialmente presentamos la declaración XML, indicando la versión y la codificación utilizada. El elemento “alumnos” hace de elemento raíz, y contiene a su vez dos elementos “alumno”. Cada uno de estos elementos tiene a su vez un atributo “identificación” y contiene otro elemento llamado “nombre”, en el cual hay contenidos datos de carácter. Cada

etiqueta se cierra al final de su contenido colocando el nombre de la etiqueta con el símbolo “/” delante.

2.1.2. SOAP

SOAP (Simple Object Access Protocol) es un protocolo cuyo objetivo es la comunicación entre aplicaciones mediante la utilización de XML. De hecho, SOAP es documentación XML propiamente dicha con un formato específico para realizar las peticiones y respuestas a servicios Web. Gracias a ello es totalmente independiente de la máquina o lenguaje que lo utilice. Además se adhiere a la programación orientada a objetos, como lo son los lenguajes Java y C# utilizados en este proyecto. Puesto que simplemente se trata de texto, no se especifica la forma de adherirse a ningún protocolo de transporte y queda a elección absoluta del desarrollador. A continuación mostramos la estructura básica de un mensaje SOAP.

```
<?xml version=”1.0”  encoding=”UTF-8”?>
<soap:Envelope xmlns:soap=”http://www.pagina.dominio/soap-envelope”>
    <soap:Header>
      ...
    </soap:Header>
    <soap:Body>
      ...
    </soap:Body>
</soap:Envelope>
```

Como podemos observar no difiere de cualquier otro documento XML. Iniciamos el documento con la declaración XML y proseguimos con el lenguaje de etiquetas. Solo que en este caso las etiquetas deben seguir este esquema para ser fieles al protocolo SOAP.

El elemento *Envelope* hace de elemento raíz, conteniendo la cabecera y el cuerpo del mensaje. El *Envelope* tiene como atributo un “*namespace*”, espacio de nombres, cuyo valor es una URL que apunta a la descripción del mensaje SOAP que va a enviarse.

Respecto a la cabecera y el cuerpo: ambos son opcionales pero si se utilizan deben estar en el orden mostrado y deben tener el mismo “*namespace*” que el *Envelope* (en este caso es “soap”). La cabecera suele contener datos relacionados con encaminamiento, o información del destino y el origen del mensaje. Por su parte, el cuerpo contiene los datos útiles del mensaje.

2.1.3. WSDL

WSDL (Web Services Description Lenguaje) es un formato mediante el cual se describen los métodos que nos ofrece un servicio Web y como emplearlos. De nuevo, se basa en esquemas XML con una estructura bien definida.

Puede emparejarse con el protocolo SOAP de forma que este utilice las funciones listadas en el WSDL de la forma indicada en el documento.

Básicamente un fichero de este tipo contiene: la URL y el espacio de nombres del servicio Web a utilizar; una lista de sus funciones; los parámetros requeridos para cada función y su tipo y, finalmente, los parámetros de retorno y su tipo. Si nos paramos a pensar un momento, son todas las variables con las que nos encontrábamos anteriormente al describir el protocolo SOAP y resulta totalmente lógico que un cliente no posea ninguno de estos datos a-priori.

Por ello estos ficheros suelen alojarse junto con el servicio Web con el que queremos comunicarnos. Mediante el siguiente esquema extraído de http://www.w3c.es/Divulgacion/Guiasbreves/ServiciosWeb quedará más clara la relación a la que nos referimos.

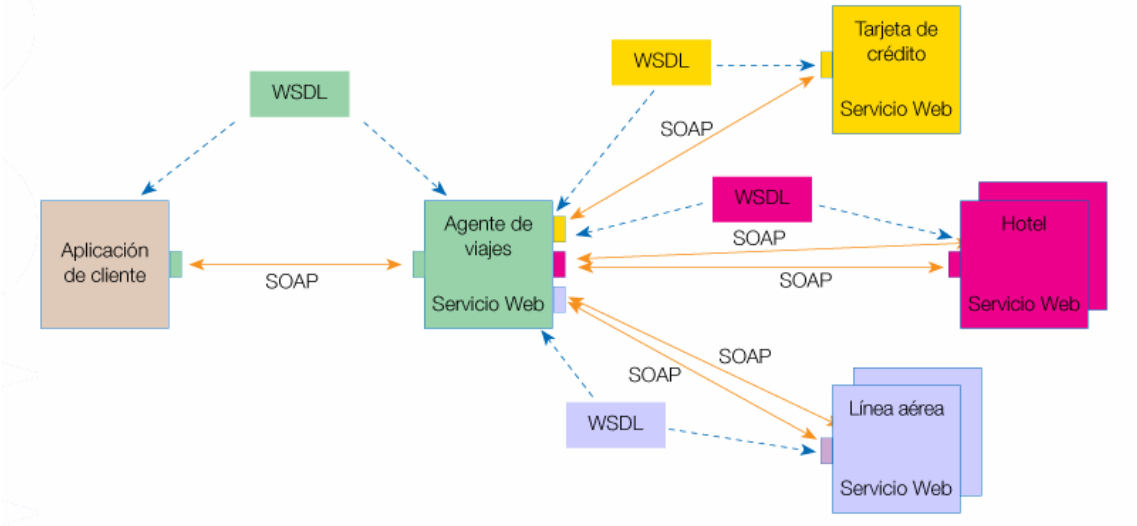

Fig 2.2 Comunicación entre servicios Web

Inicialmente se obtiene el WSDL y una vez conocemos como comunicarnos con el servicio Web, el protocolo SOAP entra en acción.

2.1.4. UDDI

UDDI (Universal Description, Discovery and Integration) es la respuesta a la pregunta que nos queda: ¿como sabe el cliente donde buscar el WSDL del servicio que desea?

UDDI es en líneas generales un registro formado en XML de los servicios Web publicados en una red, como si de unas páginas amarillas se tratara. Mediante este registro nuestro buscador sabrá de donde obtener el WSDL y, una vez sepamos como utilizar el servicio Web, hacerlo mediante SOAP.

Con estas definiciones ya deberíamos tener una idea general de cómo nuestro cliente interactúa con los servicios Web. Ahora introduciremos los conceptos de red P2P – DHT, esenciales para comprender este proyecto.

2.2. P2P-DHT

Es importante entender el concepto de P2P puesto que este proyecto se asienta sobre este tipo de redes. Una red P2P (Peer-to-Peer) tiene como característica principal que cualquier nodo puede abrir una conexión con otro para realizar una petición. A diferencia de las comunicaciones cliente-servidor donde ambas partes siempre realizan el mismo papel, en una red de Punto a Punto este rol puede cambiar, y cualquier nodo puede hacer de cliente o de servidor en cualquier momento.

Las ventajas son muchas y no es difícil deducir por qué. Si en una red cliente-servidor este último cae, nos encontraremos con que el servicio ha dejado de funcionar. Aparte, todos los clientes realizan sus peticiones al mismo servidor de forma que puede llegar a saturarse. En cambio en una red P2P no existe un servidor fijo, lo cual añade robustez al sistema y, puesto que todos los nodos pueden realizar y responder peticiones, el ancho de banda requerido se reparte entre todos, haciéndolo mucho más escalable.

Muchos programas de intercambio de ficheros utilizan este tipo de comunicaciones, como por ejemplo Kazaa y eDonkey. Existen varios tipos de redes P2P, dependiendo de la gestión de información:

- Redes centralizadas: En este tipo de redes un nodo se encarga de gestionar los intercambios de información entre peers. El problema es que se pierde la robustez característica de las redes P2P, ya que si este nodo cae no se podrán realizar nuevas búsquedas de información.

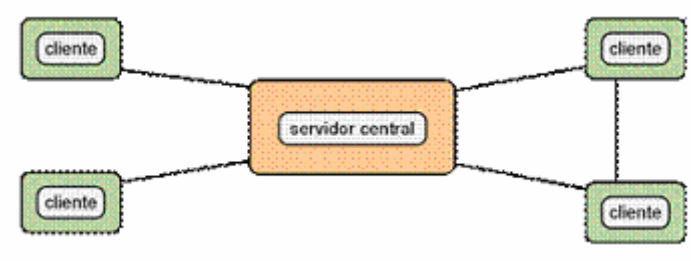

Fig 2.3 Red centralizada

- Redes descentralizadas: Este sistema tiene como ventaja que todos los nodos gestionan las transferencias. El problema es que se pierde eficiencia ya que las consultas se realizan a varios nodos en vez de a uno.

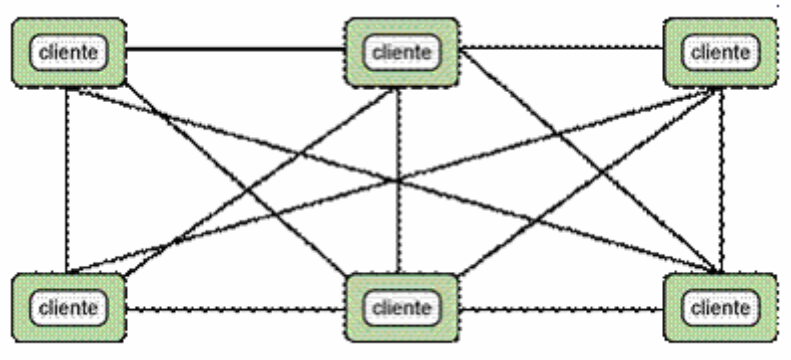

Fig 2.4 Red descentralizada

- Redes híbridas: Es el sistema más impuesto debido a que es robusto y más eficiente que las redes descentralizadas. La ventaja reside en que no todos los nodos gestionan transferencias, pero si varios proporcionalmente al número de peers en la red, de forma que no toda la responsabilidad recae en uno y las consultas no se realizan en toda la red.

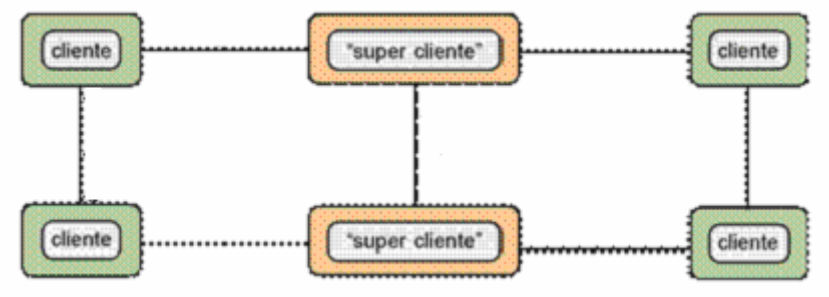

Fig 2.5 Red híbrida

En una red P2P como es la que Pastry crea, la información se almacena en tablas de hash. Por esto denominamos a estos sistemas DHT (Distributed Hash Table). Una tabla de Hash consiste en un array de elementos donde cada uno posee su correspondiente clave. De esta forma cuando añadimos un elemento a la tabla tenemos que calcular un índice a partir de la clave, intentando que sea lo mas distinto posible al resto para mejorar la eficiencia en búsquedas futuras.

Cuando realizamos una búsqueda indicamos una clave con la cual, mediante el cálculo antes mencionado, obtendremos un índice válido. Con este índice la función encargada comparará en la tabla con los índices previamente insertados, encontrará el elemento asociado y nos lo devolverá.

Las tablas de Hash son muy efectivas cuando el número de entradas es alto, como sucede en las redes P2P.

En la siguiente figura se puede observar lo anteriormente descrito: http://upload.wikimedia.org/wikipedia/commons/b/b9/Tabla_hash1.png

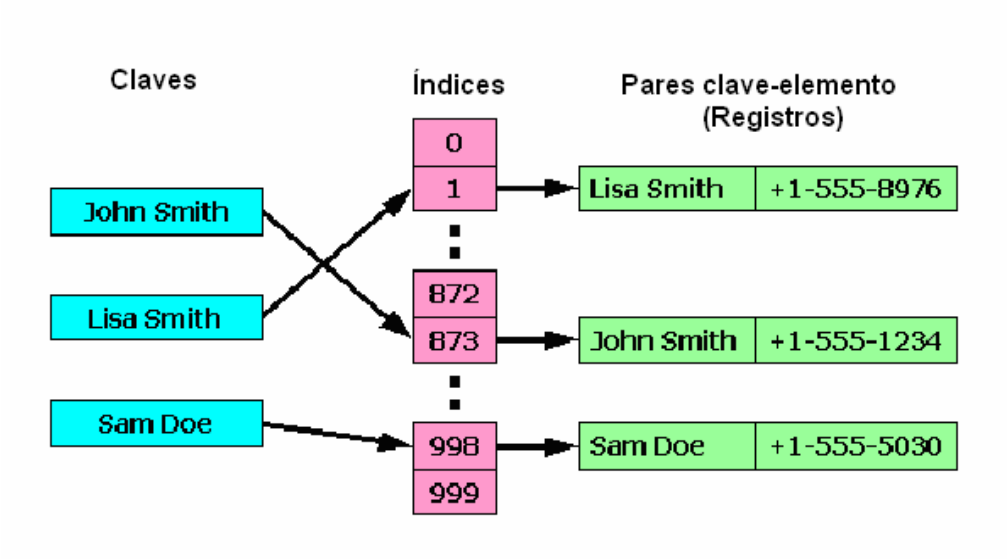

Fig 2.6 Esquema de una tabla de Hash

Hay muchos otros factores a tener en cuenta de las redes P2P, pero no profundizaremos en ellos ya que no son relevantes para el entendimiento de este proyecto.

CAPÍTULO 3: SOFTWARE UTILIZADO

Para la realización de este proyecto hemos utilizado los siguientes componentes de software:

- Lenguaje programación: Java y C#
- Creación de aplicaciones Web accesibles vía móvil: Visual Studio .NET 2005
- Ejecución de aplicaciones Java: JVM (Java Virtual Machine)
- Ejecución de aplicaciones .NET: .Net Framework
- Creación y consumo de Servicios Web: Axis
- Servidor Web: Tomcat
- Red P2P – DHT: FreePastry

3.1. Java y JVM

FreePastry esta implementado en Java, por lo que todas las modificaciones y aportaciones realizadas sobre su código se han hecho con este lenguaje.

Java es un lenguaje de programación orientado a objetos cuya sintaxis se basa en C y C++. La compilación se realiza hacia bytecode, que se trata de un código con abstracción inferior al del propio lenguaje de programación pero superior que el código máquina. Esto se hace para depender menos del hardware específico en cada caso. La ventaja de esto que es que mediante una máquina virtual como es la Maquina Virtual de Java (JVM) podemos interpretar el código sobre cualquier plataforma. Esta máquina virtual se encarga de convertir este bytecode en código máquina en el instante antes de la ejecución de forma que sea más eficiente.

3.2. C# y .NET framework

C# es otro lenguaje de programación orientado a objetos, muy similar a Java y también basado en C y C++.

Hemos escogido este lenguaje porque a diferencia de Java, C# ha sido concebido para programar aplicaciones para la plataforma .NET (pese a ser independiente de ella). Dicha plataforma facilita mucho la transparencia de redes y al igual que Java también ofrece independencia de plataformas.

Además, el entorno de desarrollo de Visual Studio .NET está creado para dar a luz aplicaciones Windows, Web y orientadas a dispositivos portátiles, lo cual nos interesaba para la realización de este proyecto.

Su funcionamiento es similar al de Java: cuando realizamos la compilación del código fuente, la salida resultante es en MSIL (Microsoft Intermediate Lenguaje), al igual que bytecode lo es para Java. Para su ejecución

necesitamos .NET Framework instalado en nuestra máquina, igual que para Java necesitábamos JVM (Java Virtual Machina). Este software será el encargado de pasar el MSIL a código máquina en el instante antes de la ejecución.

3.3. Axis

Apache – Axis es una implementación de SOAP, protocolo que hemos explicado en la sección de conceptos básicos de este proyecto. La versión que nosotros utilizaremos está implementada en Java, y nos permitirá crear nuestros servicios Web a partir de archivos .class diseñados en Java. Además podremos publicarlos utilizando nuestro servidor Web: Tomcat.

3.4. Tomcat

Apache - Tomcat es un servidor Web en el que alojaremos nuestros servicios Web. También está implementado en Java por lo que funcionará sobre cualquier máquina que tenga instalada la JVM.

3.5. FreePastry

FreePastry es una API (Application Programing Interface) de Java concebida para crear aplicaciones en comunicación P2P, de forma totalmente descentralizada y distribuida. Como hemos explicado en conceptos básicos, se trata de una red P2P – DHT, aunque presenta algunas características propias que hacen que no se trate de una red Peer - to - Peer pura.

3.5.1. Estructura de la red Pastry

La red Pastry se encarga de realizar las conexiones entre nodos y de mantenerlas de forma totalmente transparente para los usuarios. Estas conexiones se realizan en anillo, por lo que no se trata de una red Peer - to - Peer convencional.

Cuando un nodo entra en el sistema, Pastry le adjudica un identificador. Este identificador será la clave utilizada para las tablas DHT, las cuales contendrán los registros de los nodos del sistema. Como veremos mas adelante, este identificador decidirá la posición del anillo que ocupará el nuevo integrante, por lo que Pastry también se encargara de reestructurar la arquitectura del anillo.

De la misma forma, cuando un nodo abandona la red, Pastry advierte al resto de nodos y conecta entre si a los vecinos del nodo desconectado, manteniendo el anillo siempre cerrado.

Es importante tener en cuenta que cuando iniciamos un nodo, indicamos el puerto de escucha para otros, y la dirección y puerto al que vamos a

conectarnos. Si el endpoint (dirección IP + puerto) destino es un nodo perteneciente a la red Pastry, este nodo hará de Bootstrap para el nuevo incorporado. Esto significa que se conectarán para luego incorporar al nuevo a la red Pastry, y finalmente crear la tabla de rutas para el resto de nodos. Si no se encuentra ningún otro nodo, Pastry iniciará la creación de un nuevo anillo.

Más detalladamente, Pastry asigna un identificador hexadecimal a cada nodo que se incorpora a la red. Este identificador, decidido de forma totalmente aleatoria, se compone de 160 bits representados por 20 dígitos hexadecimales. El valor determinará la posición del nodo dentro del anillo, ya que Pastry comunicará inmediatamente cuales son los nodos con el identificador inmediatamente más grande y más pequeño de la red, situando al nuevo integrante en medio de ambos.

Cuando creamos un nuevo nodo, Pastry solo muestra los primeros 6 dígitos hexadecimales del identificador, simplemente porque por regla general son suficientes para hacernos una idea de donde se sitúa nuestro nodo. Ahora mostraremos unos ejemplos de cómo se reparten los nodos en función a su identificador:

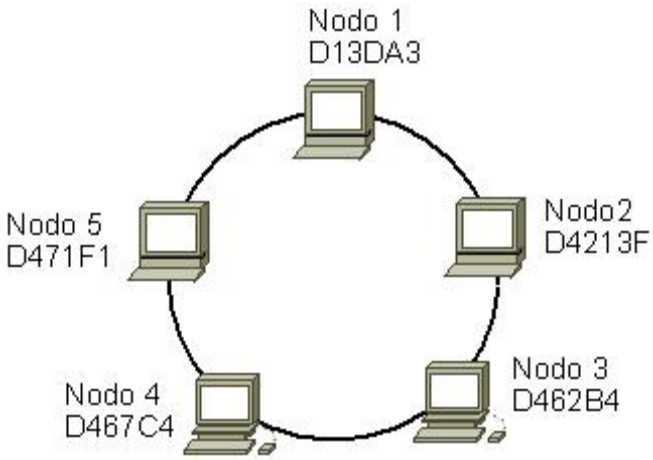

Fig 3.1 Red Pastry

Como podemos ver los nodos se han ordenado de identificador más bajo a identificador más alto, quedando finalmente el nodo con el valor más bajo y más alto con conectividad directa para cerrar el anillo. Si incluyéramos un nodo con el identificador D22C45, es deducible que se incorporará entre el nodo 1 y el nodo 2. De la misma forma, si el nodo 4 abandonara la red Pastry, el nodo 3 y el nodo 5 pasarían a estar directamente conectados.

3.5.2. Encaminamiento

En Pastry, cada nodo tiene una tabla de encaminamientos en las que mantiene la relación entre el identificador y la IP real de los nodos. De esta forma decide hacia donde encaminar la información. Cada nodo no posee siempre todos los

registros clave – dirección IP del sistema, pero si los suficientes para que los nodos cercanos sepan hacia donde encaminar el mensaje.

Entonces, cuando un mensaje va a ser enviado, el nodo emisor determina la clave más cercana respecto a la clave del nodo destino en su tabla de encaminamientos, y lo envía hacia ese nodo. En efecto, el mensaje no pasa por todos los nodos, sino que se envía directamente al más cercano al destino conocido por el emisor. El nodo intermedio realizará la misma operación, y así sucesivamente hasta dar con el destino. Con la figura mostrada a continuación podremos hacernos una idea más aproximada de lo descrito:

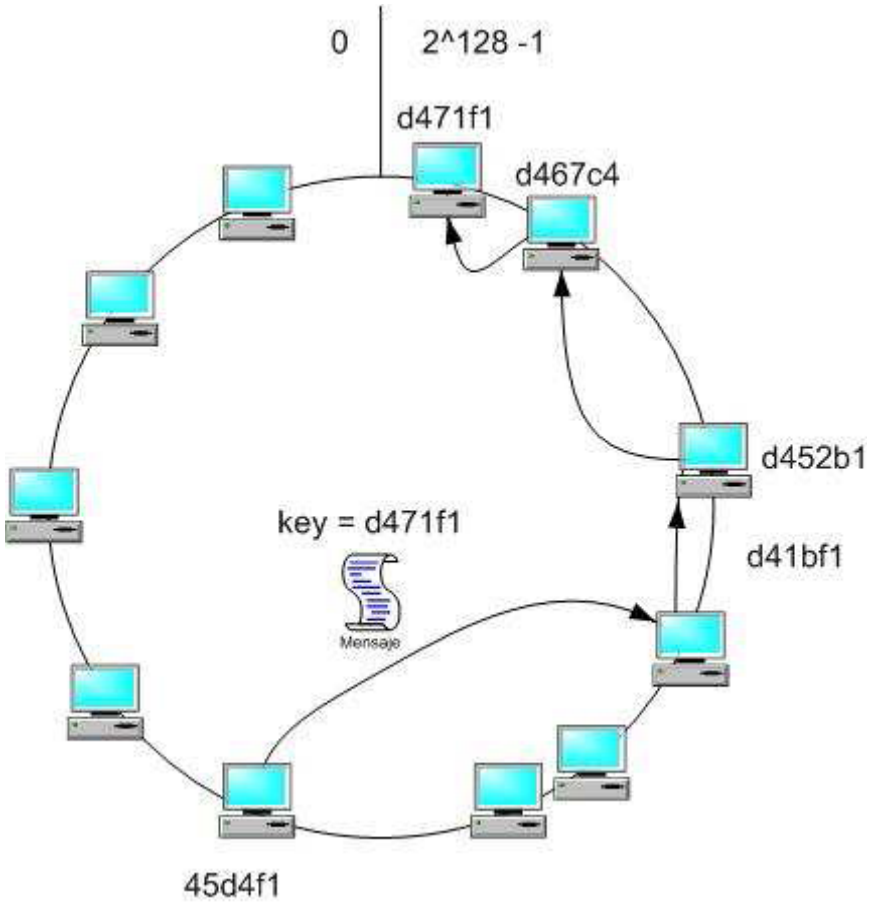

Fig 3.2 Envío de mensaje por la red Pastry

Como podemos observar, el nodo origen posee la clave del nodo destino, pero no su registro con la dirección IP por lo que no puede enviarlo directamente. Comentábamos anteriormente que los nodos poseen la clave y dirección IP de los nodos próximos. Así el nodo origen busca la clave más cercana a la del nodo destino, y le envía el mensaje. Ese nodo, más próximo al destino, sigue acercando el mensaje hacia él hasta que finalmente es entregado. Como podemos ver el origen se salta dos nodos intermedios, ya que están más alejados del destino que el nodo d41bf1, y además posee su registro en la tabla por lo que puede comunicarse con él.

CAPÍTULO 4: PREPARACIÓN DEL ENTORNO DE TRABAJO

En este capítulo explicaremos como instalar cada componente de software para poder utilizar las aplicaciones que hemos diseñado. También mencionaremos las pruebas necesarias para comprobar que lo hemos hecho bien. El proceso esta suficientemente detallado como para que cualquier usuario pueda realizarlo.

4.1. Instalación de software

Nuestro sistema es multiplataforma principalmente porque utiliza software como máquinas virtuales, que hacen que las aplicaciones no dependan del sistema operativo ni del hardware que las sostiene, sino de ellas. Pero previamente estas máquinas virtuales, así como el resto de componentes de software, deben ser instalados. Todo lo requerido es citado en este apartado.

4.1.1. JVM

Se puede descargar la JVM de la página de Sun Mycrosystems, directamente del link http://www.java.com/es/download/index.jsp . Este link es para descargar el JRE (Java Runtime Environment) versión 6, que es el que hemos utilizado para este proyecto. Este software es el que nos permitirá interpretar el bytecode, antes descrito en el apartado de software utilizado. El JRE incorpora la JVM, necesaria para ejecutar la aplicación FreePastry y los servicios Web que colgaremos mediante Apache + Axis.

4.1.2. .NET Framework

Se puede descargar de la página de Microsoft, directamente del link http://www.microsoft.com/downloads/details.aspx?displaylang=es&FamilyID=0856eacb-4362-4b0d-8edd-aab15c5e04f5 . Este link es para descargar la versión 2.0 que es la utilizada para poder ejecutar aplicaciones desarrolladas con Visual Studio .NET 2005, como son las aplicaciones Web para dispositivos portátiles.

4.1.3. Tomcat

Podemos descargar el software Apache – Tomcat desde su página oficial: http://tomcat.apache.org . Cuando ya lo tengamos instalado tendremos que levantarlo. Esto puede hacerse desde el icono que aparece abajo a la derecha de la pantalla en el caso de Windows, dando al botón “Start”; o mediante el ejecutable alojado en:

$TOMCAT_HOME/bin/tomcat5.exe (en caso de descargar la versión 5)

Si ahora entramos al link: http://localhost:8080 , debería aparecernos la pagina de bienvenida de Apache – Tomcat. Si es así, significa que ya tenemos nuestro servidor Web instalado y corriendo.

4.1.4. Axis

Podemos descargar el software de Apache – Axis desde la URL: http://ws.apache.org/axis/ . Utilizaremos la versión 1.4 escrita en Java. El archivo resultante estará comprimido por lo que tendremos que descomprimirlo.

Una vez hecho esto, copiaremos la carpeta "/axis" (dentro de "/webapps") en el directorio "$TOMCAT_HOME/webapps/". Ahora, si damos sobre el link: http://localhost:8080/axis/happyaxis.jsp debería aparecernos la "Axis Happiness Page".

Para poder utilizar nuestros servicios Web necesitaremos agregar los siguientes archivos a las variables de entorno:

- axis-ant.jar
- axis.jar
- commons-discovery.jar
- commons-logging.jar
- jaxrpc.jar
- log4j.jar
- saaj.jar
- wsdl4j.jar

Estas librerías podemos encontrarlas en el directorio siguiente: "$TOMCAT_HOME\webapps\axis\WEB-INF\lib".

4.1.5. Instalar aplicación Web PastryWebMovil

Esta aplicación Web que hemos desarrollado nos permitirá, vía móvil o mediante un navegador Web convencional, obtener información sobre los servicios publicados en Pastry. Esto es: listar los servicios actualmente publicados, obtener su descripción y, en el caso de servicios Web dedicados a móviles o navegadores corrientes, invocarlos.

Para instalar esta aplicación simplemente copiamos la carpeta que la contiene en el directorio "C:\Inetpub\wwwroot". Una vez copiada ya podemos acceder a este servicio con normalidad mediante el siguiente link: http://localhost/PastryWebMovil/ServiciosPastry.aspx .

Más adelante, en el capítulo 6, hablaremos extensamente de esta aplicación puesto que es el resultado de gran parte de este proyecto.

4.1.6. Instalar FreePastry

Realmente FreePastry es una API de Java, y por lo tanto no requiere ser instalada. Una API (Application Programming Interface) es una interfaz de

comunicación entre aplicaciones que ofrece una serie de métodos para ofrecer un grado de abstracción más alto y facilitar el desarrollo de aplicaciones complejas.

Podemos descargar este software en: http://freepastry.rice.edu . Así pues una vez descargada solo tendremos que descomprimirla para utilizarla.

4.2. Creación de un servicio Web

Ahora ya tenemos instalados los componentes que necesitamos para probar nuestra aplicación. El siguiente paso es crear un servicio Web para Axis y otro servicio Web para .NET. Primero realizaremos el de Axis, y luego el de .NET para que podamos comprobar las características de cada uno.

Primero tenemos que crear una aplicación sencilla mediante Java. En nuestro caso tenemos implementada una calculadora con las operaciones típicas de esta: suma, resta, multiplica y divide dos números. Una vez creada compilamos el archivo ".java" resultando mediante el comando "javac" que nos proporciona el SDK (Software Development Kid) de Sun. El resultado es el fichero "calculadora.class". Este fichero lo copiaremos en el directorio "$TOMCAT_HOME\webapp\axis\WEB-INF\classes".

Seguidamente, necesitamos hacer un "deploy" sobre el WSDD que nosotros mismos crearemos (Web Services Deployment Descriptor). Nuestro archivo WSDD es el siguiente:

```
<deployment xmlns="http://xml.apache.org/axis/wsdd/"
                xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">
  <!--
       Definimos el servicio Web a activar:
       Nombre y tipo de Servicio Web.
       RPC Llamadas a procedimientos remotos con ejecución síncrono
  -->
  <service name="CalculadoraWS" provider="java:RPC">

     <!-- Nombre de la clase que implementa los métodos expuestos -->
     <parameter name="className" value="Calculadora"/>

     <!-- Expone todos los métodos como visibles desde el exterior -->
     <parameter name="allowedMethods" value="*"/>
  </service>
</deployment>
```

Todos los documentos WSDD tienen esta estructura. Lo que hemos de tener en cuenta es el service name ("CalculadoraWS") y que al poner los parámetros tenemos que escribir la clase de donde extraemos el servicio Web (en este caso es "Calculadora").

Después de esto realizaremos el “deploy” para agregar este servicio Web a nuestro servidor Web. Para ello nos valdremos de algunas clases de “javax” necesarias para realizar este paso (también es necesario que se hayan agregado los ficheros .jar citados anteriormente al classpath). El comando a utilizar es:

$java org.apache.axis.client.AdminClient Calculadora.wsdd

Ahora solo tenemos que reiniciar Apache y entrar en la URL siguiente:

http://localhost:8080/axis/services/CalculadoraWS?method=multiplica&in0=3&in1=8

En este link estamos invocando el servicio Web CalculadoraWS, al cual le estamos pasando como parámetros el método a utilizar (“multiplica”) y dos valores para realizar la operación, que en este caso es in0 (cuyo valor es 3) y in1 (cuyo valor es 8). Así pues, el resultado que nos mostrará un navegador convencional es:

Fig 4.1 Respuesta de CalculadoraWS

Como podemos ver, se trata de una respuesta a un mensaje SOAP. En este caso es un “multiplicaResponse”, y nos devuelve el valor de la operación solicitada, que es “24”. Esto significa que nuestro servicio Web corre perfectamente, por lo que pasaremos a mostrar el funcionamiento de la red Pastry.

CAPÍTULO 5: APLICACIÓN EN FREEPASTRY

5.1. Aplicación en FreePastry

En este apartado explicaremos como hacemos funcionar la aplicación creada por el alumno Roger Martínez Terés y que nosotros hemos modificado para adecuarla a nuestro proyecto con servicios Web en .NET.

5.1.1. Ejecución de la aplicación

Para inicializar el anillo Pastry tenemos que ejecutar la clase “MyMain.class” mediante el comando “java”, y pasarle por parámetro el puerto de escucha, la dirección destino a la que conectarse y el puerto destino (en este orden). Así pues el comando resultante sería algo como esto:

$java com/tfc/MyMain 9000 localhost 9001

Puesto que estamos intentando conectarnos a la misma máquina, en caso de que no tengamos ninguna otra aplicación de Pastry escuchando por el puerto 9001, FreePastry creará un nuevo anillo. Si a continuación ejecutamos el siguiente comando:

$java com/tfc/MyMain 9002 localhost 9000

Puesto que tenemos un nodo escuchando en el puerto 9000, este nuevo nodo se conectará a él. Cabe destacar la gran ventaja de poder elegir el puerto por el que escuchamos, ya que eso nos ha permitido abrir más de un nodo en la misma máquina facilitando las pruebas realizadas.

5.1.2. Utilización del menú

Cada vez que realizamos el comando antes mencionado y creamos un nuevo nodo aparece el siguiente menú:

Fig 5.1 Menú principal

5.1.2.1. Publicar Web Service

Si pulsamos la primera opción la aplicación nos guiará para que introduzcamos los datos necesarios para la publicación del servicio Web. De esta forma, si queremos colgar el servicio “Calculadora” del que hablábamos anteriormente tendremos que introducir los datos de la siguiente manera:

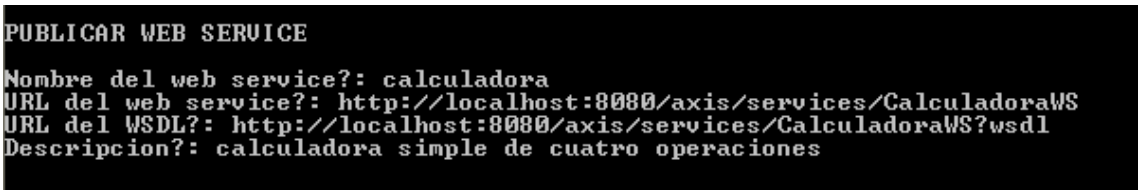

Fig 5.2 Menú Publicar Web Services

El nombre del servicio Web simplemente es una opción para facilitar al usuario su reconocimiento frente a otros servicios, y poder buscarlo posteriormente.

La URL indicará al sistema donde buscar el servicio Web a consumir. En este caso hemos puesto “localhost” cuando deberíamos poner la dirección IP de la máquina que lo aloja, para que así el resto de nodos también puedan encontrarlo. El numero “8080” es el número de puerto que utiliza Apache por defecto.

La URL del WSDL indicará al sistema los métodos que el servicio Web posee, así como las entradas que estos necesitan para funcionar y las salidas que devuelven. Todo esto ya se explicó en el “Capítulo 1: Conceptos Básicos” de este proyecto, por lo que no profundizaremos más. Para obtener el WSDL de nuestros servicios Web basta con poner la URL del mismo servicio seguida de “?wsdl”: esto genera automáticamente el archivo WSDL del servicio Web. En el ejemplo: http://localhost:8080/axis/services/CalculadoraWS?wsdl .

En la descripción simplemente introduciremos un comentario de la utilidad del servicio Web en cuestión. De nuevo es un parámetro para facilitar al usuario la identificación y consumo de las publicaciones realizadas.

Cuando se recogen todos estos datos se envía un mensaje al nodo con la ID (clave en la DHT) del nodo que tiene el servicio Web. El receptor del mensaje también comunica esta entrada a los nodos vecinos para que, en caso de caída o desconexión, no se pierda esta información. También hay un nodo especial que contiene toda la lista de servicios, la cual es comunicada a los vecinos por el mismo motivo que comentábamos anteriormente.

5.1.2.2. Listar Web Services

Esta función no requiere que el usuario introduzca ningún dato más para funcionar. Simplemente se muestra la lista de los servicios Web disponibles de la manera siguiente:

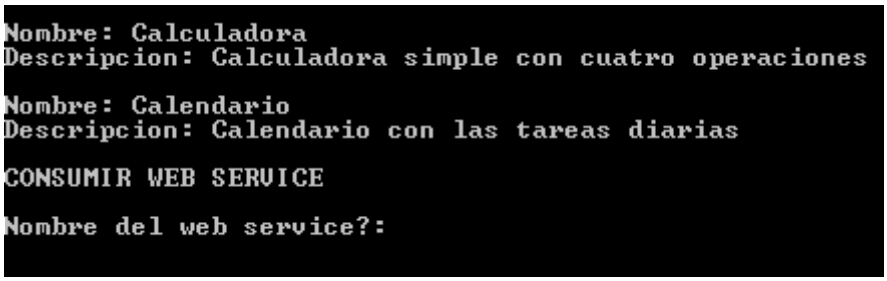

Fig 5.3 Lista de servicios Web

Inmediatamente después de listar los servicios Web, nos ofrece la opción de consumir uno de ellos.

5.1.2.3. Consumir Web Services

Para consumir un servicio Web publicado basta con poner su nombre. Automáticamente la aplicación lo buscará y mostrará sus métodos por pantalla:

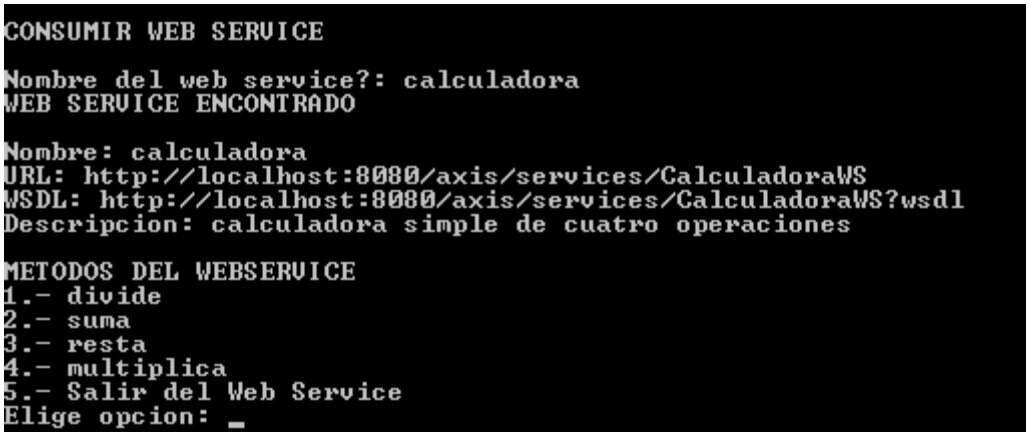

Fig 5.4 Métodos de CalculadoraWS

Esto se logra “Parseando” el documento WSDL del servicio Web de forma que conozcamos sus métodos y los parámetros necesarios para que estos funcionen. Cuando los introducimos estos se envían al propio servicio para que realice las operaciones y nos devuelva el valor mediante el protocolo SOAP.

Como ejemplo realizaremos una división para así observar el paso de parámetros de la función “divide”:

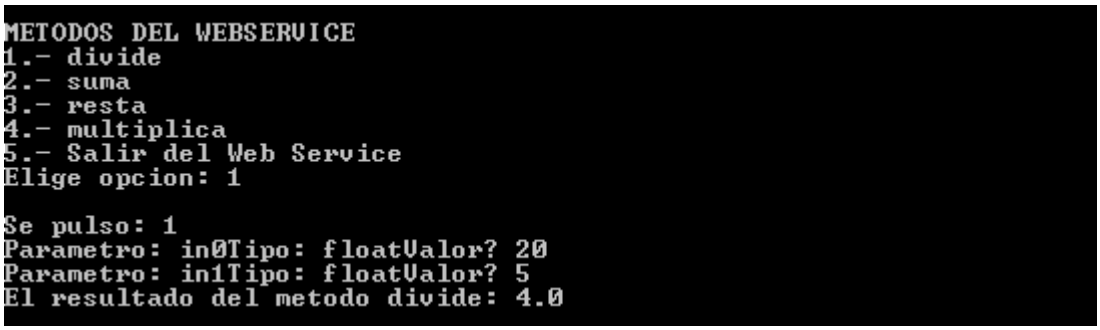

Fig 5.5 Invocación del método 1.- divide

Como podemos observar, después de elegir el método la aplicación pide que introduzcamos los parámetros requeridos. La calculadora realiza correctamente la operación demandada.

5.1.2.4. Buscar Web Service

Para buscar un servicio Web basta con introducir el nombre de éste o una parte del nombre, de forma que el sistema buscará el servicio Web cuyo nombre contenga la expresión introducida. Por ejemplo:

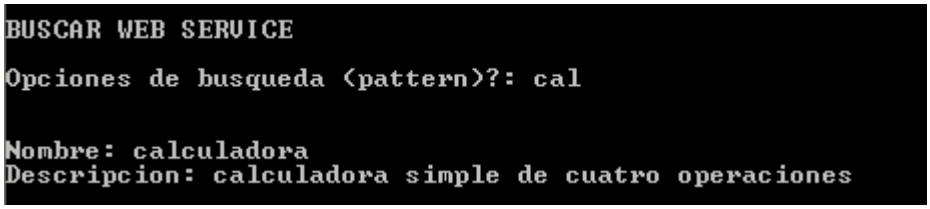

Fig 5.6 Búsqueda del servicio Web

5.1.2.5. Salir

Sale de la aplicación y desconecta el nodo. A niveles de red, este proceso se detalla en la sección “2.5.1 Estructura de la red Pastry” de este proyecto.

5.2. Mensajes en la red P2P

Nuestra aplicación implementada en Java sobre FreePastry utiliza mensajes en XML para comunicar los nodos entre ellos. Así pues, para cada operación hay un tipo de mensaje y cada mensaje tiene unos elementos concretos.

5.2.1. Mensaje de publicación

Cuando publicamos un nuevo servicio enviamos este mensaje:

```
<message>
        <type>publish</type>
        <name>Calculadora</name>
        <url>http://localhost:8080/axis/services/CalculadoraWS</url>
        <wsdl>http://localhost:8080/axis/services/CalculadoraWS?wsdl</wsdl>
        <description>Una calculadora simple</description>
</message>
```

Como vemos el elemento raíz es “message”. Esto será así en todos los mensajes de la red Pastry.

Luego definimos el “type”, elemento que también aparece en el resto de mensajes. Este elemento es utilizado por la aplicación para determinar la función que ha sido llamada y por lo tanto conocer los parámetros que se esperan.

En este caso el resto de elementos son, sencillamente, los que introducimos cuando empleamos la opción de “Publicar Web Service”.

A cada mensaje de tipo “publish” le sucede un mensaje de tipo “publishResponse” como este:

```
<message>
        <type>publishResponse</type>
        <status>ok</status>
</message>
```

El “status”, que puede ser “ok” o “ko” nos indica si la operación ha sido llevada a cabo con éxito o no.

5.2.2. Mensaje para añadir servicios a la lista

Este mensaje se envía de nodo a nodo para añadir un nuevo servicio a la lista de servicios. Cuando hacemos la petición de la lista de servicios, esta nos comunica el nombre y la descripción de estos. Por lo tanto no es difícil intuir la estructura de este tipo de mensajes:

```
<message>
        <type>add</type>
        <name>Calculadora</name>
        <description>Una calculadora simple</description>
</message>
```

5.2.3. Mensaje para pedir la lista

Este mensaje se emite cuando utilizamos la función de listar servicios Web por la línea de comandos. Entonces el nodo desde el cual realizamos la petición hace una demanda de la lista. Puesto que no requiere ningún otro elemento más que el tipo, el mensaje será así:

```
<message>
        <type>list</type>
</message>
```

El nodo que posee la lista enviará el mensaje de respuesta:

```
<message>
        <type>listResponse</type>
        <name>Calculadora</name>
        <description>Una calculadora simple</description>
        <islast>true</islast>
</message>
```

Siguiente el patrón del resto de mensajes de respuesta, el “type” es simplemente el mensaje al cual responde seguido de “Response”. El elemento “islast” informa de si es el último elemento de la lista o no. Si no lo es el que realizo la petición de lista sigue escuchando a la espera del resto de registros. Si es el último la aplicación sigue su curso.

Por otra parte, en caso de que la petición de la lista la realice la aplicación .NET PastryWebMovil, el mensaje que emitirá el nodo que realiza la petición será el siguiente:

```
<message>
        <type>list2</type>
</message>
```

Evidentemente la respuesta también será diferente, y es que esta aplicación necesita también la URL para poder redirigir al navegador hacia ella. Por eso el mensaje será como este:

```
<message>
        <type>listResponse2</type>
        <name>EncenderApagar</name>
        <description>Encender y apagar ordenadores</description>
        <url>http://localhost/EncenderApagar/MobileWebForm1.aspx</url>
        <islast>true</islast>
</message>
```

En este mensaje hemos utilizado como ejemplo un servicio Web de la plataforma .NET ya que este tipo de peticiones es útil si el servicio en cuestión puede ser visitado desde móvil, ya que la lista se le enviará a él.

5.2.4. Mensaje de búsqueda

Este mensaje se envía cuando realizamos una búsqueda de servicio Web:

```
<message>
        <type>search</type>
        <pattern>cal</pattern>
</message>
```

Llamamos “pattern” a una parte del nombre perteneciente al servicio Web que estamos buscando. Este “pattern” o fragmento será el que se compare para obtener todos los servicios que contengan ese fragmento en el interior de su nombre. En este caso enviamos “**cal**”, por lo que si tenemos un servicio Web llamado **cal**culadora y otro llamado **cal**endario, ambos serían listados. En cambio si el “pattern” fuera “**dora**”, solamente seria listado el servicio calcula**dora**.

El mensaje de respuesta es similar al de respuesta de una petición de lista, solo que en este caso se envían los mensajes que contengan el “pattern” de la forma antes mencionada.

```
<message>
        <type>searchResponse</type>
        <name>calculadora</name>
        <description>Una calculadora simple</description>
        <islast>true</islast>
</message>
```

Aquí también es necesario indicar si es el último o no puesto que, como acabamos de explicar, puede haber más de un servicio que cumpla los requisitos.

5.2.5. Mensaje de consumición

Este mensaje se lanza cuando queremos consumir un servicio Web:

```
<message>
        <type>execute</type>
        <name>Calculadora</name>
</message>
```

Simplemente comunicamos el servicio que queremos, y el sistema nos contesta con la información al completo del servicio solicitado:

```
<message>
        <type>executeResponse</type>
        <name>Calculadora</name>
        <url>http://localhost:8080/axis/services/CalculadoraWS</url>
        <wsdl>http://localhost:8080/axis/services/CalculadoraWS?wsdl</wsdl>
        <description>Una calculadora simple</description>
```

</message>

Mediante el WSDL el parser se encargará de mostrar los métodos ofrecidos por el servicio Web.

CAPÍTULO 6: DISEÑO DE LA APLICACIÓN EN .NET

En el capítulo anterior mencionábamos que habíamos realizado cambios y aportaciones en la aplicación que acabamos de explicar. En este capítulo describiremos con detalle la aplicación que hemos creado mediante Visual Studio .NET: su diseño, su funcionamiento y su objetivo.

Pero antes de empezar nos detendremos un momento para explicar porqué nos hemos decantado por este tipo de aplicaciones creadas en el entorno .NET.

6.1. Sopesando opciones

Al iniciar el proyecto teníamos claro que queríamos acceder vía móvil a servicios Web publicados en una red P2P – DHT. FreePastry nos ofrecia la red, pero aun nos quedaba por determinar como se comunicaría esta API de Java con los dispositivos portátiles.

La integración de una aplicación similar a Pastry en el móvil para que fuera tratado como un nodo más resultaba imposible, ya que el Java diseñado para dispositivos móviles esta muy limitado y no nos permitía ejecutarlo. El esquema de la idea sería éste:

Fig 6.1 Dispositivo móvil como parte de la red P2P

Descartada la idea expuesta anteriormente, pensamos en la instalación de una aplicación simple, que sencillamente realizara peticiones y esperara respuestas por parte de la aplicación en Pastry, la cual sería modificada para atender dichas peticiones. En vez de ser un nodo de la red, el móvil se dedicaría a enviarles mensajes a los nodos para pedir la información necesaria. La idea era

buena, pero existía un problema: instalar la aplicación en dispositivos de gente no experta en este campo suponía una gran pega. Este proyecto pretende ser transparente para usuarios no expertos, por lo que de nuevo la idea fue descartada. El esquema de lo que podría haber sido es el siguiente:

Fig 6.2 Dispositivo móvil con aplicación comunicándose con nodo

Finalmente llegamos a la conclusión de que el mejor método era que el móvil accediera a un servicio Web, y ese servicio Web haría de pasarela, comunicándose con la red Pastry para extraer la información necesaria. Esta idea era mucho más sencilla de llevar a cabo puesto que existen tecnologías para realizar este tipo de operaciones. Además, puesto que todos los móviles actuales tienen la capacidad de visitar servicios Web, no era necesario integrar ningún tipo de software en ellos, por lo que es totalmente transparente para el usuario y mucho más elegante. El esquema que se muestra a continuación es el esquema real de nuestro proyecto:

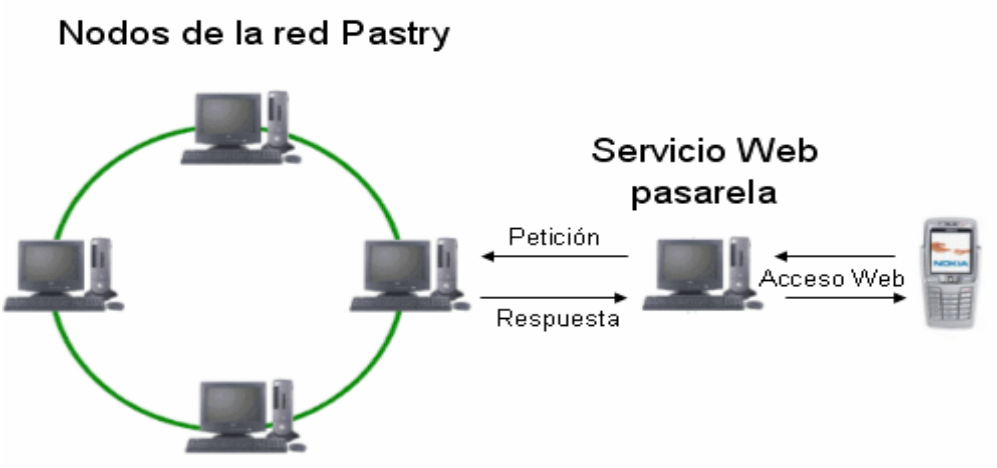

Fig 6.3 Móvil comunicándose con nodo Pastry a través de pasarela Web

Pero no llegamos directamente a la idea de implementar el servicio mediante el lenguaje C#. Estuvimos estudiando algunas otras opciones, como por ejemplo WML (Wireless Markup Lenguage). Este lenguaje es muy similar a XML, pues también esta basado en etiquetas y proviene de él. El problema es que había que editarlo manualmente o utilizar entornos no demasiado agradables.

Finalmente nos decantamos por utilizar Visual Studio .NET 2005, junto con su lenguaje C#. Este entorno esta diseñado para facilitar la creación de aplicaciones para Windows así como servicios y aplicaciones Web, tanto para buscadores convencionales como para dispositivos portátiles con sistema operativo integrado.

6.2. Aplicación Web PastryWebMovil

Para facilitar la explicación hemos dado un nombre a esta aplicación: PastryWebMovil.

Esta aplicación se trata de una aplicación Web, especialmente diseñada para su acceso mediante dispositivos portátiles.

A continuación explicaremos como se comunica esta aplicación a la cual hemos denominado PastryWebMovil.

6.2.1. Comunicación

A la aplicación que hemos implementado sobre FreePastry le hemos añadido un *thread* con un socket multicast escuchando siempre en la dirección “224.168.100.2” y en el puerto 11000. Esto significa que aunque estemos utilizando el menú para realizar alguna de las operaciones ofrecidas, siempre estará pendiente de mensajes destinados a la dirección antes mencionada.

Cuando utilizamos la aplicación PastryWebMovil, esta envía la petición multicast por toda la red. Esto significa que todos los nodos que estén escuchando la dirección antes mencionada (todos los que tienen FreePastry corriendo), reaccionarán a este mensaje. La única información de este mensaje es la dirección IP de la máquina emisora, de forma que el nodo Pastry conozca donde se aloja nuestra aplicación Web. Así evitamos tener que configurar la aplicación cada vez que la cambiamos de máquina.

Cuando uno de esos mensajes es recibido, la aplicación en FreePastry inicia una conexión TCP (una conexión segura para evitar la pérdida de mensajes) con el emisor del paquete multicast, que resulta ser la aplicación PastryWebMovil. Esta conexión se realiza sobre el puerto “8001”. Una vez realizada, se envía la lista de servicios que hay publicados actualmente, junto con su descripción y la URL del mismo.

Por cada servicio publicado se envía un mensaje, que será recibido por nuestra aplicación y, posteriormente, tratado para agregar un elemento a la tabla de servicios que posee PastryWebMovil.

Luego nuestra aplicación recorre la tabla mostrando los nombres de los servicios al usuario del móvil.

6.2.2. Utilización

Esta aplicación Web la alojaremos en el directorio “C:\Inetpub\wwwroot” para poder abrirla desde nuestro navegador. La URL sería esta:

http://localhost/PastryWebMovil/MobileWebForm1.aspx

Puesto que esta aplicación está diseñada con componentes gráficos para dispositivos portátiles, como puede ser un móvil o una PDA, es visible tanto por buscadores pertenecientes a este tipo de dispositivos como por otros buscadores. El resultado de ejecutar la URL anterior sería este:

Fig 6.4 Aplicación PastryWebMovil

Como podemos ver, su utilización es muy intuitiva y el sistema es totalmente transparente. Si pulsamos el botón “Listar Servicios Web” nos aparecerán todos los nombres de los servicios publicados.

Para demostrar el potencial del entorno de Visual Studio .Net hemos creado un servicio para el apagado y encendido de PC’s de forma remota, con el cual realizaremos los ejemplos en este apartado, pero ya detallaremos la función de este servicio más adelante.

Primero publicamos el servicio de la misma forma que lo haríamos con cualquier otro alojado en Tomcat, solo que el WSDL no es necesario ya que no tenemos que parsear las funciones, sino que la propia aplicación Web nos redirigirá al servicio Web:

```
PUBLICAR WEB SERVICE

Nombre del web service?: EncenderApagar
URL del web service?: http://localhost/WebSender/MobileWebForm1.aspx
URL del WSDL?: none
Descripcion?: Encendido y apagado remoto
```

Fig 6.5 Publicación de servicio Web .NET

Una vez publicado pulsamos el botón de “Listar Servicios Web”:

Fig 6.6 Respuesta a la petición “Listar Servicios Web”

Seleccionamos el servicio que nos interesa y pulsamos “Descripción”. El resultado de la operación sería este:

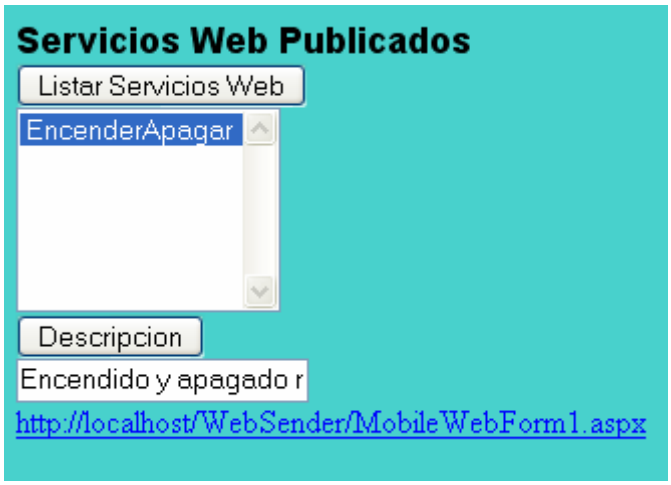

Fig 6.7 Respuesta a la petición “Descripción”

En el cuadro de texto inferior aparece la descripción del servicio Web, el cual podemos leer seleccionando el cuadro y moviéndonos a la derecha. Como podemos observar la URL del servicio aparece en cuanto pulsamos el botón “Descripción”, posibilitándonos el visitarlo con solo pulsar sobre él. Si lo hacemos directamente seremos redirigidos al servicio solicitado.

6.2.3. Servicio Web EncenderApagar

El servicio que hemos utilizado como ejemplo para la anterior explicación sirve básicamente para encender y apagar ordenadores de forma remota.

Este servicio consta de dos partes: el servicio Web que emite las órdenes y una aplicación en estado de escucha que las ejecuta. Lo hemos hecho de esta forma debido a que el sistema para apagar y encender ordenadores requiere que la red sea local y las máquinas compartan dominio. Entonces, para no limitar el alojamiento del servicio Web al interior de ese dominio (lo cual lo haría menos accesible) simplemente será necesario instalar una aplicación que hemos creado en la red, de forma que será ella la que ejecute las órdenes.

Para apagar un ordenador remotamente es necesario conocer su dirección IP de antemano, de forma que al ejecutar la orden la pasemos por parámetro. De la misma forma, para encender un ordenador se requiere la dirección física (MAC) de su tarjeta de red.

Tras sopesar las diferentes opciones para la elección de las máquinas sobre las que influir, hemos decidido que el usuario escriba en un documento llamado “Apagar.txt” las direcciones IP de los ordenadores que quiera apagar. De la misma forma, será necesario escribir en un documento llamado “Encender.txt” las direcciones físicas (MAC) de los ordenadores que quiera encender. Ambos documentos deben estar en el mismo directorio que la aplicación que los ejecuta, ya que el servicio Web EncenderApagar solo emite las órdenes que otra aplicación nuestra llevará a cabo.

La estética que se nos mostrará al visitar dicho servicio Web será el que mostramos a continuación:

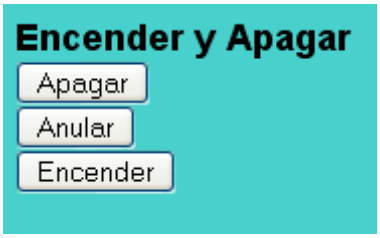

Fig 6.8 Servicio Web EncenderApagar

Cuando pulsamos uno de estos botones la aplicación Web envía una orden que será recibida por un programa en escucha constante. Este programa analizará la orden:

- Si se trata de una orden “Apagar” comprobara las direcciones IP escritas en su documento “Apagar.txt” y enviará la orden de apagado a dichas máquinas. Estas máquinas mostraran por pantalla una cuenta atrás de 30 segundos para permitir a los usuarios guardan sus trabajos. Es importante indicar que para que esto funcione dichas máquinas deben estar en el mismo dominio que la máquina emisora. Este detalle no es

demasiado importante puesto que en la mayoría de entornos donde hay suficientes computadores como para utilizar esta clase de programas suelen estar en el mismo dominio.

- Si se trata de una orden “Anular”, de nuevo se comprobaran las entradas en la lista “Apagar.txt” y anulará la orden de apagado de todas ellas. Evidentemente esta operación debe realizarse antes de que la cuenta atrás de 30 segundos expire, ya que entonces la máquina estará apagada.

- Si se trata de una orden “Encender”, el programa recogerá las entradas de MAC’s escritas en el documento “Encender.txt” y encenderá esas máquinas. A esto se le denomina Wake On LAN, ya que la señal de encendido proviene de la tarjeta de red de la máquina en cuestión.

  Para que dicha señal se envíe, la tarjeta tiene que recibir el denominado “paquete mágico” que ha de contener una estructura concreta según la dirección física que posea dicha tarjeta. Esta dirección es única para cada tarjeta, por lo que no puede haber confusión en el encendido.

Debido a que este servicio Web me pareció bastante útil, decidí implantarlo en la universidad Fundación Esade en la cual realizo las prácticas de empresa. Ahora podemos apagar y encender las 6 aulas con 20 ordenadores cada una sin tener que hacerlo manualmente.

CAPÍTULO 7: PLAN DE TRABAJO

En este apartado mostraré los procesos que he llevado a cabo para la realización de este proyecto y el tiempo aproximado empleado en cada uno de ellos. Después realizaré una especificación de costes en función al sueldo medio de un ingeniero técnico en telecomunicaciones y al material utilizado.

7.1. Tareas

Primeramente necesitaba entender las bases sobre las cuales se asienta este proyecto. Esto son las redes P2P – DHT y, más concretamente, FreePastry. Para lograrlo utilice Internet como fuente de información y estudié el tutorial de FreePastry. Calculo que esto me habrá llevado unas 100 horas.

Luego necesitaba ampliar mis conocimientos sobre tecnologías relacionadas con dispositivos portátiles. Así llegue conocer el lenguaje WML (Wireless Markup Lenguage, muy similar a XML) que, aunque finalmente no ha sido utilizado en este proyecto, si llegue a estudiar y a realizar diversos tutoriales con simuladores para móvil como OpenWave Phone Simulator. Calculo que esto me habrá llevado unas 30 horas.

Finalmente me decanté por aplicaciones surgidas de Visual Studio .NET y de su lenguage C#. Conseguí un libro de este lenguaje de programación y inicié el estudio, no solo para este proyecto sino para tener unas buenas bases en un entorno que me es agradable. Realicé muchas pruebas para dominar threats, sockets, comunicación multidifusión, interfaz gráfica, comunicaciones Web, aplicaciones para móviles, ejecución de procesos, etc…Esta ha sido la parte que más tiempo me ha llevado. Estimo que habré dedicado unas 160 horas.

Posteriormente empezó a atraerme la idea de incorporar elementos domóticos al proyecto, por lo que intente buscar dispositivos con los que realizar alguna prueba. No obstante recapacité puesto que se alejaba mucho de la idea original del proyecto y era difícil adaptarlo. Habré dedicado unas 15 horas a esta parte.

Por último recibí el proyecto de Roger Martínez Terés y me pareció atractiva la idea de retocar su programa para poder comunicarlo con una aplicación implementada en C# con las ventajas que nos brinda .NET. Así pues inicie esta tarea, para lo cual necesitaba aprender Java ya que mis conocimientos eran muy superficiales. Por suerte las similitudes que comparte con el lenguaje C# hicieron que esto no fuera un problema. Además también analicé el funcionamiento de su aplicación para poder modificarla sin errores y realicé algunas pruebas con Tomcat y Axis. Estimo que habré dedicado unas 80 horas a esta fase.

Para acabar, la redacción de esta memoria me habrá llevado unas 25 horas, y la elaboración de las diapositivas y de la exposición ante el tribunal suma otras 25 horas a este proyecto.

En total son 435 horas.

7.2. Costes asociados a este proyecto

Teniendo en cuenta que necesitamos dos ordenadores para estar completamente seguros de que el sistema funciona, y asumiendo que la duración del desarrollo de este proyecto son 435 horas aproximadamente, tenemos que:

2 ordenadores * 1000€/ordenador = 2000 €
434 horas * 10€/hora = 4340 €

Esto suma un total de 6340 euros.

CAPÍTULO 8: CONCLUSIONES

En este apartado expondré las conclusiones a las que he llegado durante la realización de este proyecto y sopesaré el resultado final con la propuesta inicial.

También propondré posibles mejoras del proyecto y las posibilidades de expansión del mismo, así como comentaré el impacto ambiental que pueda tener.

8.1. Objetivos asumidos

El objetivo inicial del proyecto era acceder a servicios Web publicados en una red P2P-DHT mediante dispositivos portátiles. Si tenemos esto en cuenta puedo afirmar que lo hemos logrado.

Así pues, viendo los resultados finales estoy satisfecho con el trabajo realizado y creo que he sido fiel a la idea inicial.

8.2. Posibles mejoras

Como se ha explicado a lo largo de este proyecto, tenemos dos formas de publicar los servicios Web: mediante Visual Studio .NET o mediante Tomcat – Axis. La aplicación que hemos realizado con .NET obtiene toda la lista de publicaciones, pero solo puede ejecutar los servicios realizados con .NET, igual que desde la línea de comandos solo podemos consumir los de Tomcat – Axis.

Estaría bien que el propio sistema solo mostrará los elementos consumibles para cada entorno. La razón de porqué no lo hemos hecho es porque simplemente no aportaba nada nuevo, y nos ha faltado tiempo.

8.3. Ampliaciones futuras

Nuestro sistema no cuenta con ninguna medida de seguridad. A la hora de acceder a los servicios Web o incluso de listarlos, podría incluirse un sistema de usuarios para otorgar permisos de visualización y consumo personalizados.

8.4. Impacto ambiental

Este proyecto es todo software. Además la capacidad de procesado que requieren las aplicaciones no es suficientemente elevada como para comprar equipos dedicados solo a ellas. Puesto que no es la causa de un incremento

significativo de gasto eléctrico ni de ningún otro tipo, el impacto ambiental de este proyecto es nulo.

8.5. Conclusiones personales

A lo largo de este proyecto me he visto a obligado a aprender mucha materia y muy variada para su realización. He aumentado mis conocimientos de C#; he profundizado en la programación con Java; me he familiarizado con XML; he estudiado WML (wireless markup lenguage) para sopesar posibilidades; he obtenido una buena perspectiva y entendimiento de las redes P2P – DHT, así como de los servicios Web y los protocolos que lo conforman; he estudiado posibles componentes de hardware para control domótico…

He tenido que tocar muchos campos para sopesar posibilidades y asimilar información muy variada, y en ese aspecto creo que este proyecto me ha aportado mucho.

CAPÍTULO 9: BIBLIOGRAFÍA

9.1. Recursos Web

Servicios Web:
http://es.wikipedia.org/wiki/Servicio_Web
http://geneura.ugr.es/~jmerelo/ws/
http://www.w3c.es/Divulgacion/Guiasbreves/ServiciosWeb

XML:
http://www.programacion.net/html/xml/htmdsssl/capitulo3/capitulo3.htm
http://www.programacion.net/html/xml/htmdsssl/capitulo4/capitulo4.htm

SOAP:
http://xml.osmosislatina.com/curso/webservices/MensajesPetroleo.htm
http://www.desarrolloweb.com/articulos/1557.php
http://www.desarrolloweb.com/articulos/1853.php
http://es.wikipedia.org/wiki/SOAP

WSDL:
http://almacen.gulic.org/diveintopython-5.4-es/soap_web_services/wsdl.html
http://es.wikipedia.org/wiki/WSDL

UDDI:
http://www.desarrolloweb.com/articulos/1599.php
http://es.wikipedia.org/wiki/UDDI

Peer-to-Peer
http://www.ub.es/geocrit/sn/sn-170-54.htm
http://es.wikipedia.org/wiki/Peer-to-peer

Hospedar un servicio Web en Tomcat - Axis
http://www.osmosislatina.com/axis/webserviceswsdl.htm

Tutorial FreePastry:
http://freepastry.rice.edu/FreePastry/tutorial

Información C# y .NET Framework:
http://msdn2.microsoft.com/en-us/default.aspx

9.2. Libros

[1] Rodríguez Gómez-Stern, M., Besteiro Gorostizaga, M.A., Desarrollo de aplicaciones .NET con Visual C#, McGraw Hill, Madrid, 2002.

ANNEXOS

TÍTULO DEL TFC: Acceso a servicios Web que han sido publicados en una red P2P DHT mediante dispositivos móviles

TITULACIÓN: Ingeniería Técnica de Telecomunicaciones, especialidad Telemática

AUTOR: Javier Olivares Sierras

DIRECTOR: Dolors Royo Vallés

Fecha: 14 de Noviembre de 2007

ANEXO 1. WSDL DEL SERVICIO CALCULADORA

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions
targetNamespace="http://localhost:8080/axis/services/CalculadoraWS"
xmlns:apachesoap="http://xml.apache.org/xml-soap"
xmlns:impl="http://localhost:8080/axis/services/CalculadoraWS"
xmlns:intf="http://localhost:8080/axis/services/CalculadoraWS"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<!--WSDL created by Apache Axis version: 1.4
Built on Apr 22, 2006 (06:55:48 PDT)-->
<wsdl:message name="divideResponse">
<wsdl:part name="divideReturn" type="xsd:int"/>
</wsdl:message>
<wsdl:message name="multiplicaRequest">
<wsdl:part name="in0" type="xsd:int"/>
<wsdl:part name="in1" type="xsd:int"/>
</wsdl:message>
<wsdl:message name="multiplicaResponse">
<wsdl:part name="multiplicaReturn" type="xsd:int"/>
</wsdl:message>
<wsdl:message name="sumaResponse">
<wsdl:part name="sumaReturn" type="xsd:int"/>
</wsdl:message>
<wsdl:message name="divideRequest">
<wsdl:part name="in0" type="xsd:int"/>
<wsdl:part name="in1" type="xsd:int"/>
</wsdl:message>
<wsdl:message name="restaRequest">
<wsdl:part name="in0" type="xsd:int"/>
<wsdl:part name="in1" type="xsd:int"/>
</wsdl:message>
<wsdl:message name="restaResponse">
<wsdl:part name="restaReturn" type="xsd:int"/>
</wsdl:message>
<wsdl:message name="sumaRequest">
<wsdl:part name="in0" type="xsd:int"/>
<wsdl:part name="in1" type="xsd:int"/>
</wsdl:message>
<wsdl:portType name="Calculadora">
<wsdl:operation name="divide" parameterOrder="in0 in1">
<wsdl:input message="impl:divideRequest" name="divideRequest"/>
<wsdl:output message="impl:divideResponse" name="divideResponse"/>
</wsdl:operation>
<wsdl:operation name="suma" parameterOrder="in0 in1">
<wsdl:input message="impl:sumaRequest" name="sumaRequest"/>
```

<wsdl:output message="impl:sumaResponse" name="sumaResponse"/>
</wsdl:operation>
<wsdl:operation name="resta" parameterOrder="in0 in1">
<wsdl:input message="impl:restaRequest" name="restaRequest"/>
<wsdl:output message="impl:restaResponse" name="restaResponse"/>
</wsdl:operation>
<wsdl:operation name="multiplica" parameterOrder="in0 in1">
<wsdl:input message="impl:multiplicaRequest" name="multiplicaRequest"/>
<wsdl:output message="impl:multiplicaResponse"
name="multiplicaResponse"/>
</wsdl:operation>
</wsdl:portType>
<wsdl:binding name="CalculadoraWSSoapBinding" type="impl:Calculadora">
<wsdlsoap:binding style="rpc"
transport="http://schemas.xmlsoap.org/soap/http"/>
<wsdl:operation name="divide">
<wsdlsoap:operation soapAction=""/>
<wsdl:input name="divideRequest">
<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://DefaultNamespace" use="encoded"/>
</wsdl:input>
<wsdl:output name="divideResponse">
<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://localhost:8080/axis/services/CalculadoraWS"
use="encoded"/>
</wsdl:output>
</wsdl:operation>
<wsdl:operation name="suma">
<wsdlsoap:operation soapAction=""/>
<wsdl:input name="sumaRequest">
<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://DefaultNamespace" use="encoded"/>
</wsdl:input>
<wsdl:output name="sumaResponse">
<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://localhost:8080/axis/services/CalculadoraWS"
use="encoded"/>
</wsdl:output>
</wsdl:operation>
<wsdl:operation name="resta">
<wsdlsoap:operation soapAction=""/>
<wsdl:input name="restaRequest">
<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://DefaultNamespace" use="encoded"/>
</wsdl:input> <wsdl:output name="restaResponse">
<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://localhost:8080/axis/services/CalculadoraWS"
use="encoded"/>
</wsdl:output>
</wsdl:operation>

<wsdl:operation name="multiplica">
<wsdlsoap:operation soapAction=""/>
<wsdl:input name="multiplicaRequest">
<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://DefaultNamespace" use="encoded"/>
</wsdl:input>
<wsdl:output name="multiplicaResponse">
<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://localhost:8080/axis/services/CalculadoraWS"
use="encoded"/>
</wsdl:output>
</wsdl:operation>
</wsdl:binding>
<wsdl:service name="CalculadoraService">
<wsdl:port binding="impl:CalculadoraWSSoapBinding"
name="CalculadoraWS">
<wsdlsoap:address
location="http://localhost:8080/axis/services/CalculadoraWS"/>
</wsdl:port>
</wsdl:service>
</wsdl:definitions>

ANEXO 2. CÓDIGO FUENTE APLICACIÓN PASTRY

2.1. MyMain.java

```
package com.tfc;

import java.io.*;
import java.util.regex.*;

import java.net.InetAddress;
import java.net.InetSocketAddress;

import rice.environment.Environment;
import rice.p2p.commonapi.Id;
import rice.p2p.commonapi.NodeHandleSet;
import rice.pastry.NodeHandle;
import rice.pastry.NodeIdFactory;
import rice.pastry.PastryNode;
import rice.pastry.PastryNodeFactory;
import rice.pastry.leafset.LeafSet;
import rice.pastry.socket.SocketPastryNodeFactory;
import rice.pastry.standard.RandomNodeIdFactory;

import java.io.*;
import java.net.*;
import java.util.*;
import java.lang.*;



public class MyMain
{

      Environment env;
      int bindport, bootport;
      InetAddress bootaddr;
      InetSocketAddress bootaddress;
      NodeIdFactory nidFactory;
      PastryNodeFactory factory;
      NodeHandle bootHandle;
      PastryNode node;
      public MyApp app;

      public void connect(int bindp, String bootaddresss, int bootp)
      {

            try
            {
                  env = new Environment();
                  bindport = bindp;
                  // build the bootaddress from the command line args
                  bootaddr = InetAddress.getByName(bootaddresss);
                  bootport = bootp;
                  bootaddress = new InetSocketAddress(bootaddr, bootp);
                  // Generate the NodeIds Randomly
                  nidFactory = new RandomNodeIdFactory(env);
```

```
                  // construct the PastryNodeFactory, this is how we
use rice.pastry.socket
                  factory = new SocketPastryNodeFactory(nidFactory,
bindport, env);

                  // This will return null if we there is no node at
that location
                  bootHandle = ((SocketPastryNodeFactory)
factory).getNodeHandle(bootaddress);

                  // construct a node, passing the null boothandle on
the first loop will cause the node to start its own ring
                  node = factory.newNode(bootHandle);

                  // the node may require sending several messages to
fully boot into the ring
                  while (!node.isReady())
                  {
                    // delay so we don't busy-wait
                    Thread.sleep(100);
                  }

                  System.out.println("Finished creating new node " +
node);

                  app = new MyApp(node);
            }
            catch (Exception e)
            {
                  System.out.println(e.toString());
            }
      }

      private void pintaMenuPrincipal()
      {
         System.out.println();
         System.out.println("MENU");
       System.out.println();
         System.out.println("1.- Publicar Web Service");
         System.out.println("2.- Listar Web Services");
         System.out.println("3.- Consumir Web Service");
         System.out.println("4.- Buscar Web Service");
         System.out.println("5.- Salir");
         System.out.println();
         System.out.print("Elije opcion: ");
      }

      private void sendHello()
      {
            LeafSet leafSet = node.getLeafSet();
            System.err.println("SIZE LEFTSET " + leafSet.maxSize());


            // this is a typical loop to cover your leafset.  Note that
if the leafset
            // overlaps, then duplicate nodes will be sent to twice
            for (int i = 1; i <= leafSet.cwSize(); i++)
            {
                  // don't send to self
                  // select the item
                  NodeHandle nh = leafSet.get(i);
```

```
                // send the message directly to the node
                app.routeMyMsgDirect(nh);

                try
                {
                      // wait a sec
                      Thread.sleep(1000);
                }
                catch (Exception e)
                {

                }

          }
     }

     private void publishWebService()
     {
        try
        {
           BufferedReader in = new BufferedReader(new
InputStreamReader(System.in));

           System.out.println();
           System.out.println("PUBLICAR WEB SERVICE");
           System.out.println();
           System.out.print("Nombre del web service?: ");
           String webservice = in.readLine();
           System.out.print("URL del web service?: ");
           String urlWebService = in.readLine();
           System.out.print("URL del WSDL?: ");
           String urlWSDL = in.readLine();
           System.out.print("Descripcion?: ");
           String desc = in.readLine();
           System.out.println();
           //System.out.println("Envio la peticion de publicar
webservice");
           System.out.println();
           app.publishWebService(webservice, urlWebService, urlWSDL,
desc);
        }
        catch (Exception e)
        {
           System.out.println(e.toString());
        }
     }

     private void searchWebService()
     {
        try
        {
           BufferedReader in = new BufferedReader(new
InputStreamReader(System.in));

           System.out.println();
           System.out.println("BUSCAR WEB SERVICE");
           System.out.println();
           System.out.print("Opciones de busqueda (pattern)?: ");
           String pattern = in.readLine();
           System.out.println();
```

```
            app.search(pattern);
         }
         catch (Exception e)
         {
            System.out.println(e.toString());
         }
      }

      public void listWebServices()
      {
            app.listWebServices();
            while (true)
            {
                  if (this.isReady())
                  {
                        this.execWebService();
                        break;
                  }
                  else
                  {
                        try
                        {
                              Thread.sleep(1);
                        }
                        catch (Exception e)
                        {
                        }
                  }
            }
      }


      private void execWebService()
      {
         try
         {
            BufferedReader in = new BufferedReader(new
InputStreamReader(System.in));

            System.out.println();
            System.out.println("CONSUMIR WEB SERVICE");
            System.out.println();
            System.out.print("Nombre del web service?: ");
            String webservice = in.readLine();
            app.executeWebService(webservice);
         }
         catch (Exception e)
         {
            System.out.println(e.toString());
         }

      }

      public boolean isReady()
      {
            return app.isReady();
      }


      public static void main(String arg[ ]) throws IOException
      {
```

```
            int port = new Integer(arg[0]);
            String adr = new String(arg[1]);
            int portd = new Integer(arg[2]);

            MyMain lect = new MyMain();

            lect.connect(port, adr, portd);
            new Escuchar_multicast(lect.app).start();

            while(true)
              {

                  if (lect.isReady())
                  {
                        BufferedReader in = new BufferedReader(new
InputStreamReader(System.in));

                        int num;
                        String str;
                        lect.pintaMenuPrincipal();



                        str = in.readLine();

                        try{
                              num = Integer.parseInt(str);
                              Pattern p = Pattern.compile("^[1-6]{1}");
                              Matcher m = p.matcher(str);
                              boolean b = m.matches();

                              if (b)
                              {
                                    switch(num)
                                    {
                                          //case 1:   lect.sendHello();
                                                      //break;
                                          case 1:
lect.publishWebService();
                                                      break;
                                          case 2:
      lect.listWebServices();
                                                      break;
                                          case 3:
      lect.execWebService();
                                                      break;
                                          case 4:
lect.searchWebService();
                                                      break;
                                          case 5:     System.exit(0);
                                                      break;
                                          default:
      System.out.println("Opcion Incorrecta");
                                                      break;
                                    }
                              }
                              else
                              {
                                    System.out.println("Opcion
Incorrecta");
```

```
                              }
                        }
                        catch (NumberFormatException nfe)
                        {
                              System.out.println("Elija un numero del 1
al 5 por favor");
                        }
                  }
                  else
                  {
                        try
                        {
                              Thread.sleep(1);
                        }
                        catch (Exception e)
                        {

                        }
                  }
            }
      }
}
```

2.2. MyApp.java

```
package com.tfc;

import com.tfc.wsdl.*;
import com.tfc.util.SimpleXML;

import rice.environment.Environment;
import rice.pastry.NodeIdFactory;
import rice.pastry.commonapi.PastryIdFactory;
import rice.pastry.PastryNode;
import rice.p2p.commonapi.Application;
import rice.p2p.commonapi.Endpoint;
import rice.p2p.commonapi.Id;
import rice.p2p.commonapi.Message;
import rice.p2p.commonapi.Node;
import rice.p2p.commonapi.NodeHandle;
import rice.p2p.commonapi.RouteMessage;
import rice.pastry.leafset.LeafSet;
import java.util.Vector;
import javax.xml.parsers.SAXParserFactory;
import javax.xml.parsers.SAXParser;
import java.io.*;
import java.util.regex.*;
import java.util.Iterator;
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
import java.net.InetAddress;
import java.net.BindException;
import java.lang.SecurityException;

/**
 * A very simple application.
```

```
 *
 * @version $Id: pretty.settings 2305 2005-03-11 20:22:33Z jeffh $
 * @author Jeff Hoye
 */
public class MyApp implements Application {
  /**
   * The Endpoint represents the underlieing node. By making calls on
the
   * Endpoint, it assures that the message will be delivered to a
MyApp on
   * whichever node the message is intended for.
   */
  protected Endpoint endpoint;

  public WebServiceP2P wsLocal;
  public WebServiceP2P wsPublished;

  private boolean isReady;
  private Id listId;
  //private String list;
  private Vector<SimpleWS> WSList;
  private PastryNode node;
      public static String IP_web;

  /**
   * Constructor for MyApp.
   *
   * @param node DESCRIBE THE PARAMETER
   */
  public MyApp(Node node) {
    // We are only going to use one instance of this application on
each PastryNode
    this.node = (PastryNode)node;
    this.endpoint = node.registerApplication(this, "myinstance");
    this.wsLocal = new WebServiceP2P();
    this.wsPublished = new WebServiceP2P();
    this.isReady=true;
    this.listId = new rice.pastry.commonapi.PastryIdFactory(new
Environment()).buildId("LlistaServeisWeb");
    //this.list="";
    this.WSList = new Vector<SimpleWS>();
  }

  public boolean isReady()
  {
      return this.isReady;
  }
  /**
   * Calld to route a publish message
   *
   * @param
   */
   public void publishWebService(String name, String url, String wsdl,
String desc)
   {
      this.isReady = false;
      WSDesc ws = new WSDesc(name, url, wsdl, desc);
            this.wsLocal.addWebServiceP2P(ws);
      //System.out.println("Creo el WSDesc");
      //Message msg = new MyMsg(endpoint.getId(), ws.id, "publish",
ws.name, ws.url, ws.wsdl, ws.description);
```

```
      Message msg = new MyMsgXML(endpoint.getId(), ws.id,
MyMsgXML.getPublishXML(ws.name, ws.url, ws.wsdl, ws.description));
      endpoint.route(ws.id, msg, null);
   }

   private void publishWebService(String name, String url, String
wsdl, String desc, boolean all)
   {
      LeafSet leafSet = this.node.getLeafSet();

        for (int i = -leafSet.cwSize(); i <= leafSet.cwSize(); i++)
      {
            if (i != 0)
            {
                  NodeHandle nh = leafSet.get(i);
                  Message msg = new MyMsgXML(endpoint.getId(),
nh.getId(), MyMsgXML.getPublish2XML(name, url, wsdl, desc));
                  endpoint.route(null, msg, nh);
            }
            }
   }

   public void executeWebService(String name)
   {
      this.isReady = false;
      Environment env = new Environment();
      PastryIdFactory localFactory = new
rice.pastry.commonapi.PastryIdFactory(env);
      Id id = localFactory.buildId(name);
      Message msg = new MyMsgXML(endpoint.getId(), id,
MyMsgXML.getExecuteXML(name));
      endpoint.route(id, msg, null);
   }

   public void listWebServices()
   {
      this.isReady = false;
      Message msg = new MyMsgXML(endpoint.getId(), this.listId,
MyMsgXML.getListXML());
        endpoint.route(this.listId, msg, null);
      }
      public void listWebServices2(String ipweb)
      {
            this.IP_web = ipweb;
            this.isReady = false;
            Message msg = new MyMsgXML(endpoint.getId(), this.listId,
MyMsgXML.getListXMLWeb());
            endpoint.route(this.listId, msg, null);
      }

   public void search(String pattern)
   {
      this.isReady = false;
      Message msg = new MyMsgXML(endpoint.getId(), this.listId,
MyMsgXML.getSearchXML(pattern));
      endpoint.route(this.listId, msg, null);
   }

  /**
   * Called to route a message to the id
   *
```

```
   * @param id DESCRIBE THE PARAMETER
   */
  public void routeMyMsg(Id id) {
    //System.out.println(this + " enviat a " + id);
    Message msg = new MyMsg(endpoint.getId(), id);
    endpoint.route(id, msg, null);
  }

  /**
   * Called to directly send a message to the nh
   *
   * @param nh DESCRIBE THE PARAMETER
   */
  public void routeMyMsgDirect(NodeHandle nh) {
    //System.out.println(this + " enviar directe a " + nh);
    Message msg = new MyMsg(endpoint.getId(), nh.getId());
    endpoint.route(null, msg, nh);
  }

   /**
   * Called when we receive a message.
   *
   * @param id DESCRIBE THE PARAMETER
   * @param message DESCRIBE THE PARAMETER
   */
  public void deliver(Id id, Message message) {
    //System.out.println(this + "id " + id + " rebut " + message);
    MyMsgXML msg = (MyMsgXML)message;

    if (SimpleXML.getValue(msg.xml, "type").equals("publish"))
    {
         //WSDesc ws = new WSDesc(msg.nameWS, msg.urlWS,
msg.urlWSDLWS, msg.descWS);
        WSDesc ws = new WSDesc(SimpleXML.getValue(msg.xml,
"name"),SimpleXML.getValue(msg.xml,"url"),SimpleXML.getValue(msg.xml,
"wsdl"),SimpleXML.getValue(msg.xml, "description"));
        this.wsPublished.addWebServiceP2P(ws);
        System.err.println("WEB SERVICE RECIBIDO");
          Message m = new MyMsgXML(endpoint.getId(), msg.from,
MyMsgXML.getPublishResponseXML("ok"));
        endpoint.route(msg.from, m, null);
        Message m2 = new MyMsgXML(endpoint.getId(), this.listId,
MyMsgXML.getAddXML(SimpleXML.getValue(msg.xml,
"name"),SimpleXML.getValue(msg.xml,"description"),
SimpleXML.getValue(msg.xml, "url")));
        endpoint.route(this.listId, m2, null);
        this.publishWebService(SimpleXML.getValue(msg.xml,
"name"),SimpleXML.getValue(msg.xml,"url"),SimpleXML.getValue(msg.xml,
"wsdl"),SimpleXML.getValue(msg.xml, "description"), true);
    }

    else if (SimpleXML.getValue(msg.xml, "type").equals("publish2"))
    {
         //WSDesc ws = new WSDesc(msg.nameWS, msg.urlWS,
msg.urlWSDLWS, msg.descWS);
        WSDesc ws = new WSDesc(SimpleXML.getValue(msg.xml,
"name"),SimpleXML.getValue(msg.xml,"url"),SimpleXML.getValue(msg.xml,
"wsdl"),SimpleXML.getValue(msg.xml, "description"));
        this.wsPublished.addWebServiceP2P(ws);
        //System.out.println("WEB SERVICE RECIBIDO");
        System.err.println("Agregado copia de web service");
```

```
   }

    else if (SimpleXML.getValue(msg.xml,
"type").equals("publishResponse"))
    {
        if (SimpleXML.getValue(msg.xml, "status").equals("ok"))
          {
            System.out.println("WEB SERVICE PUBLICADO");
        }
        else
        {
            System.out.println("ERROR: WEB SERVICE NO PUBLICADO");
          }
        this.isReady = true;
    }

    else if (SimpleXML.getValue(msg.xml, "type").equals("search"))
    {
      Vector<SimpleWS> tmp = new Vector<SimpleWS>();

      for (int i=0; i<this.WSList.size(); i++)
      {
            if
(java.util.regex.Pattern.compile(SimpleXML.getValue(msg.xml,"pattern")
).matcher(this.WSList.get(i).getName() + " " +
this.WSList.get(i).getDescription()).find())
            {
                  tmp.addElement(this.WSList.get(i));
            }
      }

      Message m;

      //Recorro el vector con los elementos encontrados
      if (tmp.isEmpty())
      {
            m = new MyMsgXML(endpoint.getId(), this.listId,
MyMsgXML.getSearchXML("none","none",true));
            endpoint.route(msg.from, m, null);
      }
      else
      {
            for (int i=0; i<tmp.size(); i++)
            {
                  //System.out.println("\r\nNombre: " +
this.WSList.get(i).getName() + "\r\nDescripcion: " +
this.WSList.get(i).getDescription());
                  if (i == tmp.size() - 1)
                  {
                        m = new MyMsgXML(endpoint.getId(), this.listId,
MyMsgXML.getSearchXML(tmp.get(i).getName(),tmp.get(i).getDescription()
,true));
                  }
                  else
                  {
                        m = new MyMsgXML(endpoint.getId(), this.listId,
MyMsgXML.getSearchXML(tmp.get(i).getName(),tmp.get(i).getDescription()
,false));
                  }
                  endpoint.route(msg.from, m, null);
            }
```

```
      }
    }

    else if(SimpleXML.getValue(msg.xml,
"type").equals("searchResponse"))
    {
      System.out.println();
      System.out.println("Nombre: " + SimpleXML.getValue(msg.xml,
"name"));
      System.out.println("Descripcion: " + SimpleXML.getValue(msg.xml,
"description"));
      if (SimpleXML.getValue(msg.xml, "islast").equals("true"))
            this.isReady = true;
    }

    else if (SimpleXML.getValue(msg.xml, "type").equals("add"))
    {
        //this.list = this.list.concat(msg.nameWS);
        this.WSList.addElement(new
SimpleWS(SimpleXML.getValue(msg.xml,"name"),SimpleXML.getValue(msg.xml
,"description"), SimpleXML.getValue(msg.xml,"url")));

        System.err.println("Yo soy la lista");

        LeafSet leafSet = this.node.getLeafSet();

          for (int i = -leafSet.cwSize(); i <= leafSet.cwSize(); i++)
        {
            if (i != 0)
            {
                  NodeHandle nh = leafSet.get(i);
                  Message m = new MyMsgXML(endpoint.getId(),
nh.getId(), MyMsgXML.getAdd2XML(SimpleXML.getValue(msg.xml, "name"),
SimpleXML.getValue(msg.xml, "description"),
SimpleXML.getValue(msg.xml, "url")));
                  endpoint.route(null, m, nh);
            }
            }
    }

    else if (SimpleXML.getValue(msg.xml, "type").equals("add2"))
    {
        //this.list = this.list.concat(msg.nameWS);
        this.WSList.addElement(new
SimpleWS(SimpleXML.getValue(msg.xml,"name"),SimpleXML.getValue(msg.xml
,"description"), SimpleXML.getValue(msg.xml, "url")));
        System.err.println("Tengo la copia de la lista");
    }

    else if (SimpleXML.getValue(msg.xml, "type").equals("execute"))
    {
        WSDesc ws =
wsPublished.getDescWSName(SimpleXML.getValue(msg.xml,"name"));

        if (ws != null)
        {
            Message m = new MyMsgXML(endpoint.getId(), msg.from,
MyMsgXML.getExecuteXML(ws.name, ws.url, ws.wsdl, ws.description));
            endpoint.route(msg.from, m, null);

        }
```

```
        else
        {
            Message m = new MyMsg(endpoint.getId(), msg.from,
MyMsgXML.getExecuteXML("none", "none", "none", "none"));
            endpoint.route(msg.from, m, null);
            //System.out.println("WEB SERVICE NO ENCONTRADO :-(");
        }
    }

    else if(SimpleXML.getValue(msg.xml,
"type").equals("executeResponse"))
    {
        if (!SimpleXML.getValue(msg.xml,"name").equals("none"))
        {
            System.out.println("WEB SERVICE ENCONTRADO");
            System.out.println();
            System.out.println("Nombre: " +
SimpleXML.getValue(msg.xml,"name"));
            System.out.println("URL: " +
SimpleXML.getValue(msg.xml,"url"));
            System.out.println("WSDL: " +
SimpleXML.getValue(msg.xml,"wsdl"));
            System.out.println("Descripcion: " +
SimpleXML.getValue(msg.xml,"description"));
            System.out.println();
            getObjectsWsdl(SimpleXML.getValue(msg.xml,"url"),
SimpleXML.getValue(msg.xml,"wsdl"),
SimpleXML.getValue(msg.xml,"name"));
        }
        else
            System.out.println("WEB SERVICE NO ENCONTRADO");
        this.isReady = true;
    }
    else if(SimpleXML.getValue(msg.xml, "type").equals("list"))
    {
      Message m;
      for (int i=0; i<this.WSList.size(); i++)
      {
            //System.out.println("\r\nNombre: " +
this.WSList.get(i).getName() + "\r\nDescripcion: " +
this.WSList.get(i).getDescription());
            if (i == this.WSList.size() - 1)
            {
                  m = new MyMsgXML(endpoint.getId(), this.listId,
MyMsgXML.getListXML(this.WSList.get(i).getName(),this.WSList.get(i).ge
tDescription(),true));
            }
            else
            {
                  m = new MyMsgXML(endpoint.getId(), this.listId,
MyMsgXML.getListXML(this.WSList.get(i).getName(),this.WSList.get(i).ge
tDescription(),false));
            }
            endpoint.route(msg.from, m, null);
      }
      //System.err.println(listws);
      //Message m = new MyMsg(endpoint.getId(), this.listId,
"listResponse", this.list);

    }
```

```
      else if (SimpleXML.getValue(msg.xml, "type").equals("list2"))
      {
            Message m;
            for (int i = 0; i < this.WSList.size(); i++)
            {
                  //System.out.println("\r\nNombre: " +
this.WSList.get(i).getName() + "\r\nDescripcion: " +
this.WSList.get(i).getDescription());
                  if (i == this.WSList.size() - 1)
                  {
                        m = new MyMsgXML(endpoint.getId(), this.listId,
MyMsgXML.getListXMLWeb(this.WSList.get(i).getName(),
this.WSList.get(i).getDescription(), this.WSList.get(i).getUrl(),
true));
                  }
                  else
                  {
                        m = new MyMsgXML(endpoint.getId(), this.listId,
MyMsgXML.getListXMLWeb(this.WSList.get(i).getName(),
this.WSList.get(i).getDescription(),
this.WSList.get(i).getUrl(),false));
                  }
                  endpoint.route(msg.from, m, null);
            }
            //System.err.println(listws);
            //Message m = new MyMsg(endpoint.getId(), this.listId,
"listResponse", this.list);

      }




    else if(SimpleXML.getValue(msg.xml,
"type").equals("listResponse"))
    {
      System.out.println();
      //MyApp.enviar_info_web(msg);
      System.out.println("Nombre: " + SimpleXML.getValue(msg.xml,
"name"));
      System.out.println("Descripcion: " + SimpleXML.getValue(msg.xml,
"description"));
      if (SimpleXML.getValue(msg.xml, "islast").equals("true"))
            this.isReady = true;
    }

      else if (SimpleXML.getValue(msg.xml,
"type").equals("listResponse2"))
      {
            //System.out.println();
            MyApp.enviar_info_web(msg);
            //System.out.println("Nombre: " +
SimpleXML.getValue(msg.xml, "name"));
            //System.out.println("Descripcion: " +
SimpleXML.getValue(msg.xml, "description"));
            //if (SimpleXML.getValue(msg.xml, "islast").equals("true"))
                  //this.isReady = true;
      }
```

```
  }
  /**
   * Called when you hear about a new neighbor. Don't worry about this
method
   * for now.
   *
   * @param handle DESCRIBE THE PARAMETER
   * @param joined DESCRIBE THE PARAMETER
   */
  public void update(NodeHandle handle, boolean joined) {
      //System.out.println("\r\nUPDATE NODEHANDLE STATUS " + joined);
  }

  /**
   * Called a message travels along your path. Don't worry about this
method for
   * now.
   *
   * @param message DESCRIBE THE PARAMETER
   * @return DESCRIBE THE RETURN VALUE
   */
  public boolean forward(RouteMessage message) {
    return true;
  }

  private void getObjectsWsdl(String endpoint, String WSDL, String WS)
  {
      Parse wsdl = new Parse();
      try
      {
            SAXParserFactory factory = SAXParserFactory.newInstance();
            factory.setNamespaceAware(true);
            SAXParser SAXParser = factory.newSAXParser();
            SAXParser.parse(WSDL, wsdl);
      }
      catch (Exception e)
      {
      }

      Vector<Wsdl> wsdlObjects = wsdl.getWSDLObjects();

      int x=wsdlObjects.size();;
      int j;

      BufferedReader in = new BufferedReader(new
InputStreamReader(System.in));

      while (true)
      {
            System.out.println("METODOS DEL WEBSERVICE");

            for(int i=0; i<x; i++)
            {
                  j=i+1;
                  //System.out.println(j + ".- " +
wsdlObjects.get(i).getMethod() + " SOAPAction=
"+wsdlObjects.get(i).getSoapActionUri());
                  System.out.println(j + ".- " +
wsdlObjects.get(i).getMethod());
            }
```

```
            x=x+1;

            System.out.println(x + ".- Salir del Web Service");
            System.out.print("Elige opcion: ");

            String str;
            int num;

            try
            {
                  str = in.readLine();
                  num = Integer.parseInt(str);
                  Pattern p = Pattern.compile("^[1-" + x + "]{1}");
                  Matcher m = p.matcher(str);
                  boolean b = m.matches();
                  System.err.println("\r\nSe pulso: " + num);
                  if (b)
                  {
                        if (num==x)
                              break;
                        else
                        {
                              int y = num-1;
                              Wsdl cli = wsdlObjects.get(y);
                              for (int i=0;
i<wsdlObjects.get(y).getInput().size(); i++)
                              {
                                    System.out.print("Parametro: " +
cli.getInput().get(i).getName() +
                                                "Tipo: " +
cli.getInput().get(i).getType() +
                                                "Valor? ");
                                    str = in.readLine();

      cli.getInput().get(i).setValue(str);
                              }

                              //Deuria tenir el objecte WSDL complert.

                              SOAPClient soap = new
SOAPClient(endpoint);
                              soap.setWsdl(cli);

                              soap.init();
                              System.out.println("El resultado del
metodo " + cli.getMethod() + ": "+soap.getResult());

                              break;
                        }
                  }
                  else
                  {
                        System.out.println("Opcion Incorrecta");
                  }
            }
            catch(Exception e)
            {
            }
      }
  }
      public static void enviar_info_web(MyMsgXML msg)
```

```
      {
            try
            {
                  Socket skCliente = new Socket(IP_web.toString(),
8001);
                  PrintStream ps = new
PrintStream(skCliente.getOutputStream());

                  //ps.print("Nombre: " + SimpleXML.getValue(msg.xml,
"name"));
                  if (SimpleXML.getValue(msg.xml,
"islast").equals("true"))
                  {
                        ps.print(SimpleXML.getValue(msg.xml, "name") +
"\t" + SimpleXML.getValue(msg.xml, "description") + "\t" +
SimpleXML.getValue(msg.xml, "url") + "\t");
                  }
                  else
                  {
                        ps.print(SimpleXML.getValue(msg.xml, "name") +
"\t" + SimpleXML.getValue(msg.xml, "description") + "\t" +
SimpleXML.getValue(msg.xml, "url") + "*");
                  }
                  System.out.println("Se ha enviado un servicio");
                  skCliente.close();
            }
            catch (Exception e)
            {
                  System.out.println(e);
            }
      }

  /**
   * DESCRIBE THE METHOD
   *
   * @return DESCRIBE THE RETURN VALUE
   */
  public String toString()
  {
    return "";
  }

}
```

2.3. MyMsgXML

```
import rice.p2p.commonapi.Id;
import rice.p2p.commonapi.Message;

/**
 * An example message.
 *
 * @version $Id: pretty.settings 2305 2005-03-11 20:22:33Z jeffh $
 * @author Jeff Hoye
 */
public class MyMsgXML implements Message {
  /**
```

```
   * Where the Message came from.
   */
  Id from;
  /**
   * Where the Message is going.
   */
  Id to;

  String xml;

  public static String getPublishXML(String name, String url, String
wsdl, String description)
  {
      return "<message>" +
             "<type>publish</type>" +
             "<name>"+name+"</name>" +
             "<url>"+url+"</url>" +
             "<wsdl>"+wsdl+"</wsdl>" +
             "<description>"+description+"</description>" +
             "</message>";
  }

  public static String getPublish2XML(String name, String url, String
wsdl, String description)
  {
      return "<message>" +
             "<type>publish2</type>" +
             "<name>"+name+"</name>" +
             "<url>"+url+"</url>" +
             "<wsdl>"+wsdl+"</wsdl>" +
             "<description>"+description+"</description>" +
             "</message>";
  }

  public static String getPublishResponseXML(String status)
  {
      return "<message>" +
             "<type>publishResponse</type>" +
             "<status>"+status+"</status>" +
             "</message>";
  }

  public static String getAddXML(String name, String description,
String url)
  {
      return "<message>" +
             "<type>add</type>" +
             "<name>"+name+"</name>" +
             "<description>"+description+"</description>" +
               "<url>" + url + "</url>" +
             "</message>";
  }

  public static String getAdd2XML(String name, String description,
String url)
  {
      return "<message>" +
             "<type>add2</type>" +
             "<name>"+name+"</name>" +
             "<description>"+description+"</description>" +
               "<url>" + url + "</url>" +
```

```
            "</message>";
  }

  public static String getListXML()
  {
      return "<message>" +
             "<type>list</type>" +
             "</message>";
  }

  public static String getListXML(String name, String description,
boolean isLast)
  {
      return "<message>" +
             "<type>listResponse</type>" +
             "<name>"+name+"</name>" +
             "<description>"+description+"</description>" +
             "<islast>" + isLast + "</islast>" +
             "</message>";
  }

      public static String getListXMLWeb()
      {
            return "<message>" +
               "<type>list2</type>" +
               "</message>";
      }
      public static String getListXMLWeb(String name, String
description, String url, boolean isLast)
      {
            return "<message>" +
               "<type>listResponse2</type>" +
               "<name>" + name + "</name>" +
               "<description>" + description + "</description>" +
               "<url>" + url + "</url>" +
               "<islast>" + isLast + "</islast>" +
               "</message>";
      }

  public static String getSearchXML(String pattern)
  {
      return "<message>" +
             "<type>search</type>" +
             "<pattern>"+pattern+"</pattern>" +
             "</message>";
  }

  public static String getSearchXML(String name, String description,
boolean isLast)
  {
      return "<message>" +
             "<type>searchResponse</type>" +
             "<name>"+name+"</name>" +
             "<description>"+description+"</description>" +
             "<islast>" + isLast + "</islast>" +
             "</message>";
  }

  public static String getExecuteXML(String name)
  {
      return "<message>" +
```

```
            "<type>execute</type>" +
            "<name>"+name+"</name>" +
            "</message>";
  }

  public static String getExecuteXML(String name, String url, String
wsdl, String description)
  {
      return "<message>" +
             "<type>executeResponse</type>" +
             "<name>"+name+"</name>" +
             "<url>"+url+"</url>" +
             "<wsdl>"+wsdl+"</wsdl>" +
             "<description>"+description+"</description>" +
             "</message>";
  }
  /**
   * Constructor.
   *
   * @param from DESCRIBE THE PARAMETER
   * @param to DESCRIBE THE PARAMETER
   */
  public MyMsgXML(Id from, Id to) {
    this.from = from;
    this.to = to;
  }

  public MyMsgXML(Id from, Id to, String xml)
  {
    this.from = from;
    this.to = to;
    this.xml = xml;
  }

  /**
   * Use low priority to prevent interference with overlay maintenance
traffic.
   *
   * @return The Priority value
   */
  public int getPriority() {
    return Message.LOW_PRIORITY;
  }

  /**
   * DESCRIBE THE METHOD
   *
   * @return DESCRIBE THE RETURN VALUE
   */
  public String toString() {
    return "";
  }
}
```

2.4. WebServiceP2P.java

```
package com.tfc;
```

```
import java.util.Vector;
import java.lang.String;
import rice.environment.Environment;
import rice.p2p.commonapi.Id;
import rice.pastry.NodeIdFactory;
import rice.pastry.commonapi.PastryIdFactory;

/**
 * Class for publishing/unpublishing web services in Pastry
 *
 * @author Dolors Royo
 */
class WSDesc{
  String name;
  String description;
  String url;
  String wsdl;
  Id id;

  public WSDesc(String name, String url, String wsdl, String
description){
      this.name = name;
      this.url = url;
      this.wsdl = wsdl;
      this.description = description;
      Environment env = new Environment();
      PastryIdFactory localFactory = new
rice.pastry.commonapi.PastryIdFactory(env);
      this.id = localFactory.buildId(name);
      //System.out.println("New Web Service object cretaed "+id);
  }
}

public class WebServiceP2P{
  public Vector<WSDesc> WSDescV;


  public WebServiceP2P(){
      WSDescV = new Vector<WSDesc>();
  }

  public void addWebServiceP2P(WSDesc ws) {
      WSDescV.addElement(ws);
  }

  public void getDescWSId(Id id) {
  }

  public WSDesc getDescWSName(String name) {
      for (int i=0; i< WSDescV.size(); i++){
            WSDesc ws = WSDescV.get(i);
            if (ws.name.equals(name)){
                  return ws;
            }
      }
      return null;
  }

}
```

2.5. SOAPClient.java

```
package com.tfc;

import org.apache.axis.client.Call;
import org.apache.axis.client.Service;
import org.apache.axis.encoding.XMLType;
import javax.xml.rpc.ParameterMode;
import com.tfc.wsdl.*;
import java.util.Vector;

public class SOAPClient {
	String endpoint;
	Service service;
	Call call;
	Wsdl wsdl;
	String result;

	public SOAPClient()
	{

	}
	public SOAPClient(String ep)
	{
		this.endpoint = ep;
	}

	public void setWsdl(Wsdl ws)
	{
		this.wsdl = ws;
	}

	public Wsdl getWsdl()
	{
		return this.wsdl;
	}

	public String getResult()
	{
		return this.result;
	}

	public void init()
	{
		//this.endpoint =
"http://localhost:8080/axis/services/CalculadoraWS";

		try
		{
			this.service = new Service();
			this.call = (Call) this.service.createCall();
			this.call.setTargetEndpointAddress(new
java.net.URL(this.endpoint));
			this.call.setOperationName(this.wsdl.getMethod());
			Vector<Part> input = this.wsdl.getInput();

			int cont = 0;
			for (int i=0; i<input.size(); i++)
			{
				if (input.get(i).getType().equals("int"))
```

```
                        {

      this.call.addParameter(input.get(i).getName(), XMLType.XSD_INT,
ParameterMode.IN);
                        }
                        else if
(input.get(i).getType().equals("string"))
                        {

      this.call.addParameter(input.get(i).getName(),
XMLType.XSD_STRING, ParameterMode.IN);
                        }
                        else if
(input.get(i).getType().equals("float"))
                        {

      this.call.addParameter(input.get(i).getName(),
XMLType.XSD_FLOAT, ParameterMode.IN);
                        }
                        cont = i;
                  }

                  Part output = this.wsdl.getOutput();

                  if (output.getType().equals("int"))
                  {
                        this.call.setReturnType(XMLType.XSD_INT);
                  }

                  else if (output.getType().equals("string"))
                  {
                        this.call.setReturnType(XMLType.XSD_STRING);
                  }

                  else if (output.getType().equals("float"))
                  {
                        this.call.setReturnType(XMLType.XSD_FLOAT);
                  }

                  Object[] ob = new Object[cont+1];
                  for (int j=0; j<=cont; j++)
                  {
                        if (input.get(j).getType().equals("int"))
                        {
                              ob[j] = new
Integer(input.get(j).getValue());
                        }
                        else if
(input.get(j).getType().equals("string"))
                        {
                              ob[j] = new
String(input.get(j).getValue());
                        }
                        else if
(input.get(j).getType().equals("float"))
                        {
                              ob[j] = new
Float(input.get(j).getValue());
                        }
                  }
```

```
                if (output.getType().equals("int"))
                {
                      Integer res = (Integer) this.call.invoke(ob);
                      this.result = res.toString();
                }
                else if (output.getType().equals("string"))
                {
                      String res = (String) this.call.invoke(ob);
                      this.result = res;
                }
                else if (output.getType().equals("float"))
                {
                      Float res = (Float) this.call.invoke(ob);
                      this.result = res.toString();
                }
          }
          catch(Exception e)
          {
                e.printStackTrace();
          }

    }

}
```

2.6. Parse.java

```
package com.tfc.wsdl;

import javax.xml.parsers.SAXParserFactory;
import javax.xml.parsers.SAXParser;
import org.xml.sax.helpers.DefaultHandler;
import org.xml.sax.*;
import org.xml.sax.helpers.*;
import java.util.Vector;

public class Parse extends DefaultHandler
{

    private String url;
    private Vector<Message> messages = new Vector<Message>();
    private boolean inPortType = false;
    private Operation op;
    private PortType porttype;
    private Part part;
    private Message msg;

    public void Parse ()
    {
          System.out.println("Constructor");
    }

    public void setUrl (String url)
    {
          this.url = url;
    }

    public String getUrl ()
    {
```

```
            return this.url;
      }

      public PortType getPortType()
      {
            return this.porttype;
      }

      public Vector<Message> getMessages()
      {
            return this.messages;
      }

      public void startElement(String uri, String localName, String
qName, Attributes attributes) throws SAXException
      {
            /*if (qName.substring(qName.length()-
7,qName.length()).equals("address") &&
attributes.getLocalName(0).equals("location"))
            {
                  setUrl(attributes.getValue(0));
            }
            */
            if (qName.substring(qName.length()-
9,qName.length()).equals("operation") && inPortType)
            {
                  op = new Operation();
                  op.setName(attributes.getValue(0));
            }
            else if (qName.substring(qName.length()-
8,qName.length()).equals("portType"))
            {
                  inPortType = true;
                  porttype = new PortType();
                  porttype.setName(attributes.getValue(0));
            }
            else if (qName.substring(qName.length()-
5,qName.length()).equals("input") && inPortType)
            {

      op.setInput(attributes.getValue(0).substring(attributes.getValue
(0).indexOf(":") + 1, attributes.getValue(0).length()));
            }
            else if (qName.substring(qName.length()-
6,qName.length()).equals("output") && inPortType)
            {

      op.setOutput(attributes.getValue(0).substring(attributes.getValu
e(0).indexOf(":") + 1, attributes.getValue(0).length()));
            }
            else if (qName.substring(qName.length()-
7,qName.length()).equals("message"))
            {
                  msg = new Message();
                  msg.setName(attributes.getValue(0));
            }
            else if (qName.substring(qName.length()-
4,qName.length()).equals("part"))
            {
                  part = new Part();
```

```
      part.setName(attributes.getValue(0).substring(attributes.getValu
e(0).indexOf(":") + 1, attributes.getValue(0).length()));

      part.setType(attributes.getValue(1).substring(attributes.getValu
e(1).indexOf(":") + 1, attributes.getValue(1).length()));
            }


      }

      public void endElement(String uri, String localName, String
qName) throws SAXException
      {

            //System.out.println("Final de etiqueta: " + qName);

            if (qName.substring(qName.length()-
8,qName.length()).equals("portType"))
            {
                  inPortType = false;
            }
            else if (qName.substring(qName.length()-
9,qName.length()).equals("operation") && inPortType)
            {
                  this.porttype.addOperation(op);
            }
            else if (qName.substring(qName.length()-
7,qName.length()).equals("message"))
            {
                  this.messages.addElement(msg);
            }
            else if (qName.substring(qName.length()-
4,qName.length()).equals("part"))
            {
                  this.msg.addPart(part);
            }


      }

      public Vector<Wsdl> getWSDLObjects()
      {
            Vector<Wsdl> vWsdl = new Vector<Wsdl>();

            Vector<Operation> operations =
this.porttype.getOperations();

            for(int i=0; i<operations.size(); i++)
            {
                  Wsdl oWsdl = new Wsdl();
                  oWsdl.setMethod(operations.get(i).getName());

                  String iOp = operations.get(i).getInput();
                  String oOp = operations.get(i).getOutput();

                  for (int j=0; j<this.messages.size(); j++)
                  {
                        if (iOp.equals(this.messages.get(j).getName()))
                        {
```

```
                              Vector<Part> parts =
this.messages.get(j).getParts();
                              for(int k=0; k<parts.size(); k++)
                              {
                                    oWsdl.addInput(parts.get(k));
                              }
                        }
                        if (oOp.equals(this.messages.get(j).getName()))
                        {
                              Vector<Part> parts =
this.messages.get(j).getParts();
                              for(int k=0; k<parts.size(); k++)
                              {
                                    oWsdl.setOutput(parts.get(k));
                              }
                        }

                  }
                  vWsdl.addElement(oWsdl);
            }
            return vWsdl;
      }
}
```

2.7. Escuchar_multicast.java

```
package com.tfc;

import java.io.*;
import java.util.regex.*;

import java.net.InetAddress;
import java.net.InetSocketAddress;

import rice.environment.Environment;
import rice.p2p.commonapi.Id;
import rice.p2p.commonapi.NodeHandleSet;
import rice.pastry.NodeHandle;
import rice.pastry.NodeIdFactory;
import rice.pastry.PastryNode;
import rice.pastry.PastryNodeFactory;
import rice.pastry.leafset.LeafSet;
import rice.pastry.socket.SocketPastryNodeFactory;
import rice.pastry.standard.RandomNodeIdFactory;

import java.io.*;
import java.net.*;
import java.util.*;
import java.lang.*;

public class Escuchar_multicast extends Thread
{
      public static MyApp lect;

      public Escuchar_multicast(MyApp lector)
      {
            this.lect = lector;
      }
```

```
	public static void escuchar_multicast() throws IOException
	{
		try
		{
			byte[] buf = new byte[256];
			DatagramPacket packet = new DatagramPacket(buf,
buf.length);
			MulticastSocket socket = new MulticastSocket(11000);

	socket.joinGroup(InetAddress.getByName("224.168.100.2"));
			while (true)
			{
				socket.receive(packet);

	//System.out.println(packet.getAddress().getHostAddress());

	lect.listWebServices2(packet.getAddress().getHostAddress());
			}
		}
		catch (IOException e)
		{
		}
	}
	public void run()
	{
		try
		{
			escuchar_multicast();
		}
		catch(IOException e)
		{
		}
	}

}
```

ANEXO 3. CÓDIGO FUENTE PASTRYWEBMOVIL

3.1. MobileWebForm1.aspx.cs

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.Mobile;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.MobileControls;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;
using System.Collections.Generic;
using System.Text;
using System.Runtime.Serialization;
using System.Runtime.Serialization.Formatters.Binary;
using System.IO;
using System.Net;
using System.Net.Sockets;

namespace PastryWebMovil
{
      /// <summary>
      /// Descripción breve de MobileWebForm1.
      /// </summary>
    ///
    public struct ServicioWeb
    {
        public char[] nombre;
        public char[] descripcion;
        public char[] url;
        public int tamaño_nombre;
        public int tamaño_descripcion;
        public int tamaño_url;

        public void Inicializar()
        {
            tamaño_descripcion = 0;
            tamaño_nombre = 0;
            tamaño_url = 0;
            nombre = new char[100];
            descripcion = new char[100];
            url = new char[100];

        }
    }

      public partial class MobileWebForm1 :
System.Web.UI.MobileControls.MobilePage
      {
        static IPHostEntry hostInfo =
Dns.GetHostByName(Dns.GetHostName());
```

```
            static public IPAddress ip_servidor =
hostInfo.AddressList[0];
        static TcpListener server = new TcpListener(ip_servidor,
8001);
        static TcpClient client = null;
        MulticastSearch PastryFinder = new MulticastSearch();
        static int numero_servicios = 0;
        static bool final_lista = false;
        static private ServicioWeb[] lista = new ServicioWeb[100];

        public MobileWebForm1()
        {

        }
        protected void Page_Load(object sender, System.EventArgs e)
            {

            }

            #region Código generado por el Diseñador de Web Forms
            override protected void OnInit(EventArgs e)
            {
                  //
                  // CODEGEN: llamada requerida por el Diseñador de Web
Forms ASP.NET.
                  //
                  InitializeComponent();
                  base.OnInit(e);
            }

            /// <summary>
            /// Método necesario para admitir el Diseñador. No se puede
modificar
            /// el contenido del método con el editor de código.
            /// </summary>
            private void InitializeComponent()
            {

            }
            #endregion
        protected void Listar_btn_Click(object sender, EventArgs e)
        {
            PastryFinder.JoinMulticastGroup();
            PastryFinder.BroadcastMessage();
            String servicio;
            Respuesta.Items.Clear();
            numero_servicios = 0;
            final_lista = false;
            while (final_lista == false)
            {
                servicio = Escuchar_lista();
                lista[numero_servicios] = Añadir_Servicios(servicio);
                numero_servicios++;
            }

            //Lista_servicios.Text = lista[0].tamaño_url.ToString();

            for (int i = 0; i < numero_servicios; i++)
            {
                for (int j = lista[i].tamaño_nombre; j <
lista[i].nombre.Length; j++)
```

```
                {
                    lista[i].nombre[j] = '\t';
                }
                for (int j = lista[i].tamaño_descripcion; j <
lista[i].descripcion.Length; j++)
                {
                    lista[i].descripcion[j] = '\t';
                }
                for (int j = lista[i].tamaño_url; j <
lista[i].url.Length; j++)
                {
                    lista[i].url[j] = '\t';
                }
            }

            String aux;

            for (int i = 0; i < numero_servicios; i++)
            {
                aux = new string(lista[i].nombre);
                Respuesta.Items.Add(aux);
            }


            // Shutdown and end connection
            client.Close();
            server.Stop();
        }
        protected void Lista_servicios_TextChanged(object sender,
EventArgs e)
        {

        }

        public static String Escuchar_lista()
        {

                server.Start();
                Byte[] bytes = new Byte[256];  //buffer de lectura
                String data = null;

                //Bucle de escucha
                while (true)
                {
                    // Aceptar clientes
                    client = server.AcceptTcpClient();
                    data = null;

                    // Crear objeto stream de lectura y escritura
                    NetworkStream stream = client.GetStream();

                    int i;

                    // Bucle para recibir todos los datos del cliente
                    while ((i = stream.Read(bytes, 0, bytes.Length))
!= 0)
                    {
                        // Traducir todos los bytes de datos a ASCII
strings
                        data =
System.Text.Encoding.ASCII.GetString(bytes, 0, i);
```

```
                    }
                    return data;
                }

        }

        public static ServicioWeb Añadir_Servicios(String servicio)
        {
            bool nombre = false;
            bool descripcion = false;
            bool url = false;
            int j,i;
            ServicioWeb lista = new ServicioWeb();
            lista.Inicializar();

            for (i=0,j=0; i < servicio.Length ; i++)
            {
                if (nombre == false && servicio[i] != '\t')
                {
                    lista.nombre[j] = servicio[i];
                    j++;
                }
                else if (nombre == false && servicio[i] == '\t')
                {
                    lista.tamaño_nombre = j;
                    nombre = true;
                    j = 0;
                }
                else if (nombre == true && descripcion == false &&
servicio[i] != '\t')
                {
                    lista.descripcion[j] = servicio[i];
                    j++;
                }
                else if (nombre == true && descripcion == false &&
servicio[i] == '\t')
                {

                    lista.tamaño_descripcion = j;
                    descripcion = true;
                    j = 0;
                }
                else if (nombre == true && descripcion == true && url
== false && servicio[i] != '\t' && servicio[i] != '*')
                {
                    lista.url[j] = servicio[i];
                    j++;
                }
                else if (nombre == true && descripcion == true && url
== false && (servicio[i] == '\t' || servicio[i] == '*') )
                {
                    lista.tamaño_url = j;
                    url = true;
                    j = 0;
                }
            }
            if (servicio[servicio.Length - 1] == '\t')
            {
                final_lista = true;
            }
            return lista;
```

```
        }

        /*public static ServicioWeb[] Preparar_listado(ServicioWeb[]
lista)
        {
            int i, j;
            for (i=0; i < numero_servicios; i++)
            {
                for (j = lista[i].tamaño_nombre; j <
lista[i].nombre.Length; j++)
                {
                    lista[i].nombre[j] = '\r';
                }
                for (j = lista[i].tamaño_descripcion; j <
lista[i].nombre.Length; j++)
                {
                    lista[i].descripcion[j] = '\r';
                }
                for (j = lista[i].tamaño_url; j <
lista[i].nombre.Length; j++)
                {
                    lista[i].url[j] = '\r';
                }
            }
            return lista;
        }*/

        protected void Lista_servicios_TextChanged1(object sender,
EventArgs e)
        {

        }
        protected void Respuesta_ItemCommand(object sender,
ListCommandEventArgs e)
        {

        }
        protected void Respuesta_SelectedIndexChanged1(object sender,
EventArgs e)
        {

        }
        protected void Invocar_btn_Click(object sender, EventArgs e)
        {
        }
        protected void Respuesta_SelectedIndexChanged(object sender,
EventArgs e)
        {

        }
        protected void Descripcion_btn_Click(object sender, EventArgs
e)
        {
            char[] aux = new char[100];
            bool encontrado = false;
            for (int i = 0; i < numero_servicios; i++)
            {
                for (int j = 0; j < lista[i].tamaño_nombre; j++)
                {
                    if (lista[i].nombre[j] ==
Respuesta.Items[Respuesta.SelectedIndex].ToString().ToCharArray()[j])
```

```
                    {
                        encontrado = true;
                    }

                    else
                    {
                        encontrado = false;
                        break;
                    }
                }

                if (encontrado == true)
                {
                    string descripcion_selected = new
string(lista[i].descripcion);
                    Lista_servicios.Text=descripcion_selected;
                    string url_selected = new string(lista[i].url);
                    Link.NavigateUrl = url_selected;
                    break;
                }
            }

            //string prueba = new string(lista[1].descripcion);
            //Lista_servicios.Text =
Respuesta.Items[Respuesta.SelectedIndex].ToString();
            //String prueba = new String(lista[1].nombre);
            //Lista_servicios.Text = prueba;
                /*{
                    if
(char.IsControl(Respuesta.Selection.ToString().ToCharArray()[i]))
                    {
                        aux[i] = '\t';
                    }
                    else
                    {
                        aux[i] =
Respuesta.Selection.ToString().ToCharArray()[i];
                    }
                }
                string prueba = new string(aux);
                Lista_servicios.Text = prueba;*/

        }
    }
}
```

3.2. MulticastSearch.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using System.Net;
using System.Net.Sockets;
```

```
using System.Text;
using System.Runtime.Serialization;
using System.Runtime.Serialization.Formatters.Binary;
using System.IO;
using System.Diagnostics;

namespace PastryWebMovil
{
    /// <summary>
    /// Summary description for MulticastSearch
    /// </summary>
    public class MulticastSearch
    {
        static IPAddress mcastAddress;
        static int mcastPort;
        static Socket mcastSocket;
        public IPAddress localIPAddr;
        public MulticastSearch()
        {
            //
            // TODO: Add constructor logic here
            //
        }

        public void JoinMulticastGroup()
        {
            try
            {

                mcastSocket = new Socket(AddressFamily.InterNetwork,
SocketType.Dgram, ProtocolType.Udp);
                mcastAddress = IPAddress.Parse("224.168.100.2");
                mcastPort = 11000;

                IPHostEntry hostInfo =
Dns.GetHostEntry(Dns.GetHostName());
                localIPAddr = hostInfo.AddressList[0];

                IPEndPoint IPlocal = new IPEndPoint(localIPAddr, 0);


                mcastSocket.Bind(IPlocal);

                MulticastOption mcastOption;
                mcastOption = new MulticastOption(mcastAddress,
localIPAddr);

                mcastSocket.SetSocketOption(SocketOptionLevel.IP,
SocketOptionName.AddMembership, mcastOption);



            }
            catch (Exception e)
            {
                //Console.WriteLine("\n" + e.ToString());
            }

        }

        public void BroadcastMessage()
```

```
        {
            IPEndPoint endPoint;
            IFormatter f = new BinaryFormatter();
            MemoryStream stream = new MemoryStream();
            string message = localIPAddr.ToString();
            try
            {
                endPoint = new IPEndPoint(mcastAddress, mcastPort);
                f.Serialize(stream, message);
                byte[] b = stream.ToArray();
                stream.Close();
                mcastSocket.SendTo(b, b.Length, SocketFlags.None,
endPoint);

            }
            catch (Exception e)
            {
                //Console.WriteLine("\n" + e.ToString());
            }

            mcastSocket.Close();
        }
    }
}
```

ANEXO 4. CÓDIGO FUENTE ENCENDERAPAGAR

4.1. MobileWebForm1.aspx.cs

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.Mobile;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.MobileControls;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.Runtime.Serialization;
using System.Runtime.Serialization.Formatters.Binary;
using System.IO;
using System.Diagnostics;

namespace WebSender
{
      /// <summary>
      /// Descripción breve de MobileWebForm1.
      /// </summary>
      public partial class Web_Sender :
System.Web.UI.MobileControls.MobilePage
      {
        static IPAddress mcastAddress;
        static int mcastPort;
        static Socket mcastSocket;

            protected void Page_Load(object sender, System.EventArgs e)
            {
                  // Introducir aquí el código de usuario para
inicializar la página
            }

            #region Código generado por el Diseñador de Web Forms
            override protected void OnInit(EventArgs e)
            {
                  //
                  // CODEGEN: llamada requerida por el Diseñador de Web
Forms ASP.NET.
                  //
                  InitializeComponent();
                  base.OnInit(e);
            }

            /// <summary>
            /// Método necesario para admitir el Diseñador. No se puede
modificar
            /// el contenido del método con el editor de código.
            /// </summary>
```

```
            private void InitializeComponent()
            {

            }
            #endregion

            protected void Form1_Activate(object sender,
System.EventArgs e)
            {

            }

            protected void Apagar_btn_Click(object sender,
System.EventArgs e)
            {

            string texto_mensaje = "Apagar";
            JoinMulticastGroup();
            BroadcastMessage(texto_mensaje);

            }

            protected void Anular_btn_Click(object sender,
System.EventArgs e)
            {

                  string texto_mensaje = "Anular";
            JoinMulticastGroup();
            BroadcastMessage(texto_mensaje);

            }

            static void JoinMulticastGroup()
        {
              try
              {

                mcastSocket = new Socket(AddressFamily.InterNetwork,
SocketType.Dgram, ProtocolType.Udp);
                mcastAddress = IPAddress.Parse("225.168.100.2");
                mcastPort = 10000;

                IPHostEntry hostInfo =
Dns.GetHostEntry(Dns.GetHostName());
                IPAddress localIPAddr = hostInfo.AddressList[0];

                IPEndPoint IPlocal = new IPEndPoint(localIPAddr, 0);


                mcastSocket.Bind(IPlocal);

                MulticastOption mcastOption;
                mcastOption = new MulticastOption(mcastAddress,
localIPAddr);

                mcastSocket.SetSocketOption(SocketOptionLevel.IP,
SocketOptionName.AddMembership,mcastOption);



              }
```

```
              catch (Exception e)
              {
                //Console.WriteLine("\n" + e.ToString());
              }

            }

        static void BroadcastMessage(string message)
        {
            IPEndPoint endPoint;
            IFormatter f = new BinaryFormatter();
            MemoryStream stream = new MemoryStream();
            try
            {
                endPoint = new IPEndPoint(mcastAddress, mcastPort);
                f.Serialize(stream, message);
                byte[] b = stream.ToArray();
                stream.Close();
                mcastSocket.SendTo(b, b.Length,
SocketFlags.None,endPoint);
            }
            catch (Exception e)
            {
                //Console.WriteLine("\n" + e.ToString());
            }

            mcastSocket.Close();
        }
        protected void Command1_Click(object sender, EventArgs e)
        {
            string texto_mensaje = "Encender";
            JoinMulticastGroup();
            BroadcastMessage(texto_mensaje);
        }
}
}
```

4.2. TestMulticastOption.cs

```
using System;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.Runtime.Serialization;
using System.Runtime.Serialization.Formatters.Binary;
using System.IO;
using System.Diagnostics;
using ASOCKETLib;


namespace Mssc.TransportProtocols.Utilities
{

  public class TestMulticastOption
  {

    private static IPAddress mcastAddress;
    private static int mcastPort;
    private static Socket mcastSocket;
```

```
    private static MulticastOption mcastOption;
    private static string[] direcciones = new string[500];
    public static int elementos;

    private static void StartMulticast()
    {

      try
      {
        mcastSocket = new Socket(AddressFamily.InterNetwork,
                                 SocketType.Dgram,
                                 ProtocolType.Udp);

        //Console.Write("Enter the local IP address: ");

        IPHostEntry hostInfo =
System.Net.Dns.GetHostEntry(System.Net.Dns.GetHostName());
        IPAddress localIPAddr = hostInfo.AddressList[0];

        //IPAddress localIP = IPAddress.Any;
        EndPoint localEP = (EndPoint)new IPEndPoint(localIPAddr,
mcastPort);

        mcastSocket.Bind(localEP);


        // Define a MulticastOption object specifying the multicast
group
        // address and the local IPAddress.
        // The multicast group address is the same as the address used
by the server.
        mcastOption = new MulticastOption(mcastAddress, localIPAddr);

        mcastSocket.SetSocketOption(SocketOptionLevel.IP,
                                    SocketOptionName.AddMembership,
                                    mcastOption);

      }

      catch (Exception e)
      {
        Console.WriteLine(e.ToString());
      }
    }

    private static string ReceiveBroadcastMessages()
    {

      byte[] bytes = new Byte[100];
      EndPoint groupEP = (EndPoint) new IPEndPoint(mcastAddress,
mcastPort);
      //EndPoint remoteEP = (EndPoint) new
IPEndPoint(IPAddress.Any,5000);
      MemoryStream stream1 = new MemoryStream(bytes);
      IFormatter f = new BinaryFormatter();
      string md;

      try
      {
          mcastSocket.ReceiveFrom(bytes, ref groupEP);
          md = (string)f.Deserialize(stream1);
```

```
          stream1.Close();
          mcastSocket.Close();
          return md;
      }

      catch (Exception e)
      {
        Console.WriteLine(e.ToString());
        return e.ToString();
      }
    }

      private static void Apagar()
      {
          int i;
          direcciones = leer_fichero_direcciones("Apagar.txt");
          for (i = 0; i <= elementos; i++)
          {
              Process.Start(@"C:\WINDOWS\system32\shutdown.exe", "-s -
m " + direcciones[i]);
          }
      }

      private static void Anular()
      {
          int i;
          direcciones = leer_fichero_direcciones("Apagar.txt");
          for (i = 0; i <= elementos; i++)
          {
              Process.Start(@"C:\WINDOWS\system32\shutdown.exe", "-a -
m " + direcciones[i]);
          }
      }

      private static void decidir_accion(string mensaje)
      {
          if (mensaje == "Apagar")
          {
              Apagar();
          }

          else if (mensaje == "Anular")
          {
              Anular();
          }

          else if (mensaje == "Encender")
          {
              Encender();
          }
      }

      private static string[] leer_fichero_direcciones(string fichero)
      {
          StreamReader lector = new StreamReader(fichero);
          string[] direcciones = new string[500];
          elementos = 0;
          int i = 0;
          while (!lector.EndOfStream)
          {
              direcciones[i] = lector.ReadLine();
```

```
            i++;
            elementos++;
        }
        lector.Close();
        return direcciones;
    }

    private static void Encender()
    {
        int i;
        WOL objWol = new WOLClass();
        objWol.Clear();

        direcciones = leer_fichero_direcciones("Encender.txt");
        for (i = 0; i < direcciones.Length; i++)
        {
            objWol.WakeUp(direcciones[i]);
        }

    }

  public static void Main(String[] args)
  {
    // Initialize the multicast address group and multicast port.
    // Both address and port are selected from the allowed sets as
    // defined in the related RFC documents. These are the same
    // as the values used by the sender.
    mcastAddress = IPAddress.Parse("225.168.100.2");
    mcastPort = 10000;
    string mensaje;

    while (true)
    {// Start a multicast group.
        StartMulticast();

        // Receive broadcast messages.
        mensaje = ReceiveBroadcastMessages();
        decidir_accion(mensaje);
    }
  }
 }
}
```